The Report Committee for Prabhat Bhattarai
certifies that this is the approved version of the following report:

Monte Carlo Glauber Model: Quadrupole Correlations in Au+Au Collisions

APPROVED BY

SUPERVISING COMMITTEE:

Stephen G Walker, Supervisor

Robert Ray, Co-Supervisor

Monte Carlo Glauber Model: Quadrupole Correlations in Au+Au Collisions

by

Prabhat Bhattarai, M.S.; M.S.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN STATISTICS

The University of Texas at Austin

December 2015

To my family and my mentors

Acknowledgments

I would like to thank all who have supported and encouraged me to accomplish this project. I am especially thankful to my wife, Sharmila. Thank you very much for your support in every step of the way. I am very grateful to have parents who encouraged me to pursue my goals.

I am grateful to Prof. Stephen G Walker and Dr. Robert L. Ray for the wonderful guidance. Thank you for guiding me in every step. I also thank Jerry Hoffmann for supporting my interest of pursing my degree in Statistics. I am lucky to have worked with great colleagues such as Christina Markert, Jo Schambach, Deepa Thomas, Alex Jentsch, Justin Blair and Erin Gauger.

I would like to thank Kumar Mainali for sparking my interest in this field.

Prabhat Bhattarai

Monte Carlo Glauber Model: Quadrupole Correlations in Au+Au Collisions

Prabhat Bhattarai, M.S. Stat.
The University of Texas at Austin, 2015

Supervisor: Stephen G Walker
Co-Supervisor: Robert Ray

The study of quarks and their interaction through gluons has been active area of research since its discovery. It has been about two decades that Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory is dedicated to study interaction between quarks by producing nuclear matter in extremely dense and hot environment. It has been understood that colliding beams of atomic nuclei at a speed close to the speed of light creates the hot and dense environment in which all quarks in the nuclei de-confine to form a short-lived state of matter called Quark Gluon Plasma (QGP). Because of the short life time of QGP, the only way to study such matter is through the final state of particles. A significant feature of the final state distribution of particles is an azimuthal anisotropy which is dominated by the second Fourier component; the amplitude is proportional to parameter v_2. One of the major interpretations of this anisotropy is based on the hypothesized thermal equilibrium of the QGP leading to pressure driven collective flow. The other is

that quantum interference among many quark and gluon scatterings leads directly to anisotropy in the final state. The present report presents a way to calculate the observed quadrupole correlation amplitude, v_2, without assuming collective flow. The study uses a Monte Carlo method to simulate the gold-gold nuclear collision data using experimental results from proton-proton collisions. The quality of simulated results is assured by comparison to theoretical understanding of the phenomenon as well as to the experimental data. This report presents studies of two-particle correlations, whose derivation can be traced back to Pearson's correlation coefficient, in azimuthal angular space of the simulated tracks of the particles produced in the gold-gold of collisions. The correlation result is fitted to extract the v_2 and compared to the same quantity from the experimental data. The comparison suggests that a fraction of the v_2 in gold-gold collisions can be accounted for by phenomenon not associated with collective flow.

Table of Contents

List of Figures

Chapter 1

Introduction

Nucleons (protons and neutrons) consist of quarks and gluons. Relativistic heavy-ion physics provides an opportunity to measure internal structure of nucleons. Furthermore, it provides a unique opportunity to produce a phase of matter called the Quark Gluon Plasma (QGP). In the QGP the quarks and gluons, which are bound together inside the nucleons by the strong force (one of the four forces in nature: Gravity, electromagnetic, weak and strong), are free to move over distances larger than the size of nucleon. The study of the QGP is interesting because it might be similar to the universe immediately (μs) after the Big Bang.

In the relativistic heavy-ion collisions, nuclei of atoms such as copper, gold or lead etc, are collided with each other at a speed close to the speed of light. A single collision is called an *event* and information from millions of collisions are collected in the experiments. The energy of such collision is very high such that the confined quarks and gluons inside the nucleons are briefly

de-confined. Shortly after the de-confinement, quarks and gluons recombine to form particles such as pions, kaons, protons etc. which are observed in the detectors. In heavy ion collision experiments, the detectors are centered around the heavy ion beam crossing point where the events occur. The detectors are capable of measuring the momentum, energy and velocity of the particles. The important kinematic and dynamic properties of the state of matter produced right after the collision must be inferred through the study of these observables. The evolution of a relativistic heavy-ion collision is shown schematically in Figure 1.1.

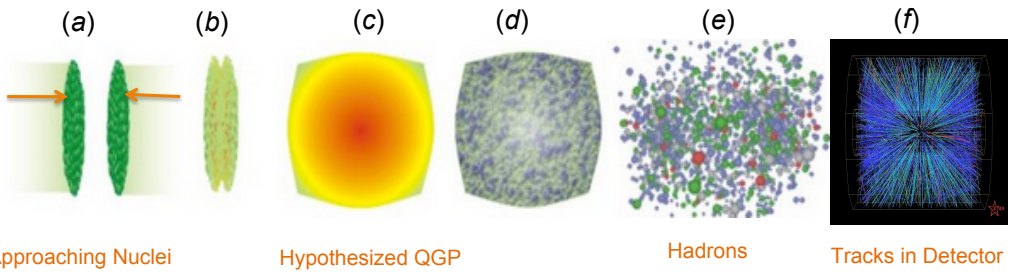


Figure 1.1 A schematic diagram of a relativistic heavy-ion collision. (a) Two nuclei are shown as thin disks due to relativistic effect called Lorentz contraction. (b) The state in which the two nuclei just start interacting. (c) Formation of QGP. (d) The QGP is thermalized. (e) Formation of particles such as pion, kaons, protons etc. (f). The reconstructed tracks in the detectors.

In the relativistic collision of heavy-ions, such as gold + gold ions, a single event may produce number of final-state particles in the range from 2 to few thousand. The collision is called *peripheral* if the colliding nuclei just touch each other. In a peripheral collision the distance between the colliding nuclei is close to $2R$ where R is the radius of a nucleus. Similarly, the collision

is called *central* is the colliding nuclei completely overlap each other during the collision. In a central collision, the distance between centers the of two nuclei is approximately 0. The degree of overlap a collision is called *impact parameter*. The number of particles produced in a collision is called event *multiplicity*. The multiplicity of a collision is a function of centrality and energy of the collision. The multiplicity is higher if the collision energy (speed) is higher. Similarly, in a head-on collision between ions, the multiplicity is highest where as in peripheral collisions it is lowest. In the experiment measurements are limited by $efficiency$ and *acceptance*, coverage area, of the detectors. The detectors are optimized for maximum efficiency and acceptance at minimum cost.

In heavy-ion collisions, we can't directly measure some of the important initial geometry variables such as impact parameter (b)- the degree of nuclear overlap during a collision, and reaction plane angle (ψ)- azimuthal orientation of nuclei prior to the collision. The Monte Carlo Glauber (MCG) model simulates these initial geometry parameters and has been shown to successfully describe many of the the particle production observables in relativistic heavy-ion collisions. The MCG model uses initial collision geometry to generate particle distributions after the collision event-by-event[2] [3].

The scope of this report covers a simulations the heavy-ion collision data and its analysis. The simulation is based on the results from proton+proton collisions at center of mass energy of 200 GeV. The simulation is carried out for gold+gold collisions at center of mass energy of 200 GeV.

The analysis method consists of finding correlation between two particles, *two particle correlations*, in collection of similar events.

A heavy-ion nucleus contains many nucleons. Thus some important physics in a heavy-ion collision is expected to come from a nucleon nucleon collision. In the present study the azimuthal anisotropy parameter v_2 in gold+gold collisions is simulated by summing the measured v_2 in proton+proton collisions form many random nucleon+nucleon collisions. In Chapter 2 the details of the Monte Carlo Glauber Model are presented. The two particle angular correlation is defined in Chapter 3. The results of the study and discussions are presented in Chapter 4 and 5.

The Monte Carlo Glauber Model has been coded in C++ and in an object oriented language called ROOT. The following steps summarize the algorithm of the code.

- Randomly generate the distribution of nucleons in each colliding nuclei based on the theoretical understanding of spatial distributions of nucleons in the nucleus.

- For a randomly chosen impact parameter-distance between the centers of two colliding nuclei, calculate the distance between two nucleons in the projectile and the target nuclei. If the distance is smaller than a threshold (based on classical scattering theory), then record this as a colliding pair of nucleons.

- Each collision of nucleons is likely to produce particles. Generate these particles based on experimental results for the distribution of particles from nucleon nucleon collisions (proton+proton collisions).

- Assign the azimuthal direction of the produced particles based on the measured anisotropy in proto+proton collisions.

- Repeat the process for each collision between nucleon pairs in gold+gold collision. Note that the current study was designed for gold+gold collisions. So there are 179 nucleons in each nucleus. The maximum number of collisions between two nucleons could go up to 179×179.

- Record the azimuthal angle of each produced particle for each event. In the current study, 10 million events were studied.

- Define a correlation measure and find the two particle correlations using the azimuthal direction of the particles.

The main part of the code that generates the events is included in Appendix 1. However, because it was not the goal of this project, the part of the code dedicated to computing correlations from generated events was included in this article. The code for computing correlations is closely related to the standard correlation code present in the software library for the STAR experiment based at Brookhaven Laboratory.

The computation process was challenging because of the time required to generate events and finding two particle correlations. Generation of each

event could take up to a half second. To generate several million events could take up to tens of hours. Therefore, the Texas Advanced Computing Center (TACC) facility was used to run the simulation in parallel environment. Events generated in each parallel node were combined together to get the final correlation results.

Chapter 2

Monte Carlo Glauber Model

In heavy-ion collisions, initial geometric quantities such as impact parameter and directionality of the collision cannot be directly determined experimentally. The Monte Carlo Glauber Model has been successful to simulate the collision and estimate some of the important physics of the collision. In the following chapter, the detail of the simulations is presented.

2.1 Monte Carlo Glauber Model

Monte Carlo Glauber Model is a geometric approach to build up a realistic nuclear collision model. As a first step of the model calculation, the position of each nucleon (proton or neutron) in a nucleus is determined according to Woods-Saxon probability distribution function [2]. The Woods-Saxon distribution gives three dimensional density profile of nucleons in the nucleus. The density of nucleons at any distance r from center of a nucleus is:

$$\rho(r) = \rho_0 \frac{1 + \omega(r/R)^2}{1 + e^{\frac{r-R}{a}}}, \tag{2.1}$$

where ρ_0 denotes the density of nucleons in the center of nucleus of radius R, ω produces non-uniform densities in the nuclear interior and a denotes skin depth. For Gold, ^{179}Au, R= 6.38 fm, a =0.535 fm and ω =0 [3]. Since the differential volume element at distance r from the center of sphere is $4\pi r^2 dr$, the radius of randomly chosen nucleon of a nucleus comes from the sampling distribution $4\pi r^2 \rho(r)$.

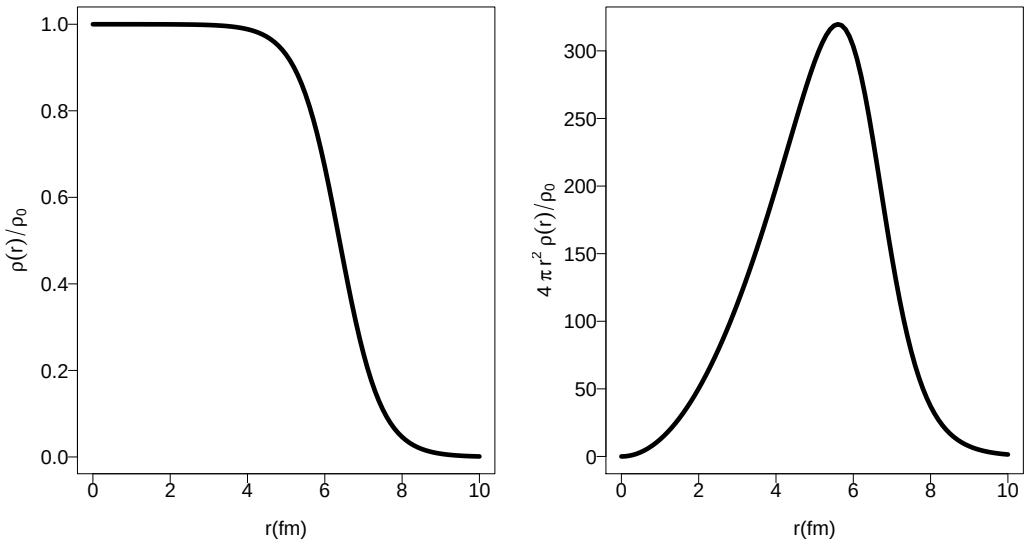


Figure 2.1 Nuclear density profile. Left panel: nuclear density profile as function of radius. Right panel: profile of number of nucleons as a function of radius. Density of the nucleus decreases as radius increases. However, because of increasing surface area, the number of nucleons tends to increases up to certain magnitude of radius.

Further, we need azimuthal (ϕ) and polar(θ) angles of nucleons with

respect to nuclear axes. Both of the azimuthal and polar angles are randomly assigned. For random numbers $x_1, x_2 \in [0, 1]$, the following transformation gives the angles in spherical polar coordinates system:

$$\phi = 2\pi x_1 \tag{2.2}$$

and

$$\theta = arccos(2x_2 - 1), \tag{2.3}$$

where $(0 \leq \phi \leq 2\pi)$ is distributed uniformly in entire azimuth whereas $(0 \leq \theta \leq \pi)$ and is distributed such that it is maximum around equatorial plane and becomes zero around both poles. Figure 2.2 shows the distribution of r, θ, and ϕ in an event sample.

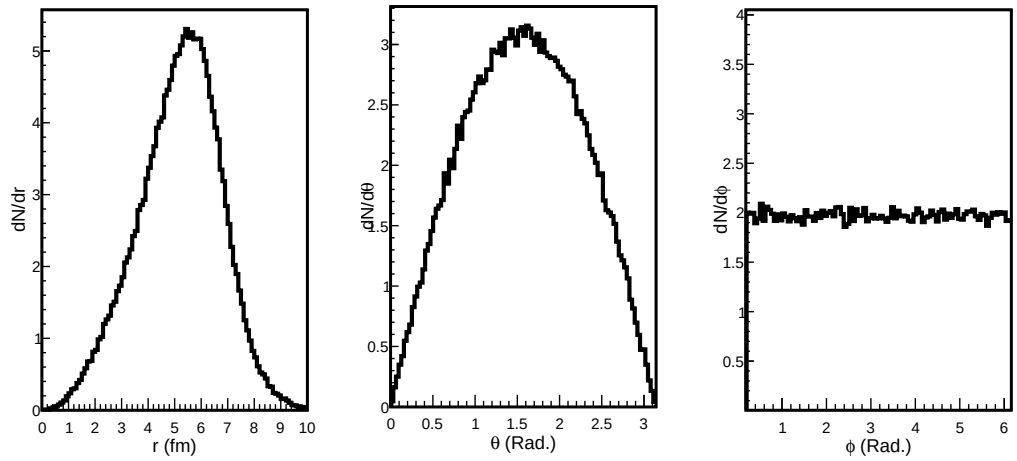


Figure 2.2 From left to right: distribution of position(r), polar angle (θ) and azimuth (ϕ) normalized to one event. The data from this small statistical sample verifies that the distributions follow the theoretical expectations.

With given distribution functions of three coordinates, a nucleus can be simulated. The next important parameter for a collision is the impact parameter. As two nuclei approach each other, the distance between the centers of the nuclei in transverse plane is called impact parameter (b). The collision cross-section is proportional to b.

$$\frac{d\sigma}{db} = 2\pi b, \tag{2.4}$$

where σ is the collision cross-section. As the collision changes from head on to the most peripheral, the value of b changes from 0 to $2R$.

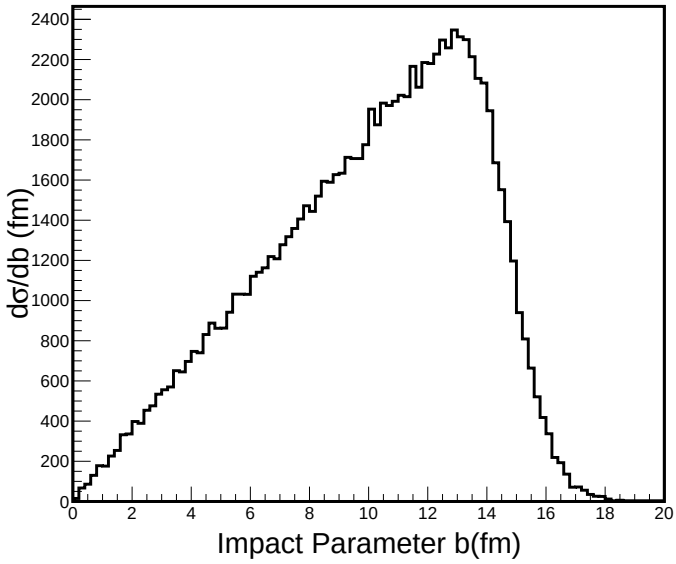


Figure 2.3 Total geometric cross-section MCG calculation. Cross-section for head-on collision is 0 and it grows linearly up-to 14 fm. Because of the fall in density of nucleons at larger r, the cross-section decays to zero rapidly. The yield in the y-axis is obtained from 100,000 collisions with non-zero cross-section.

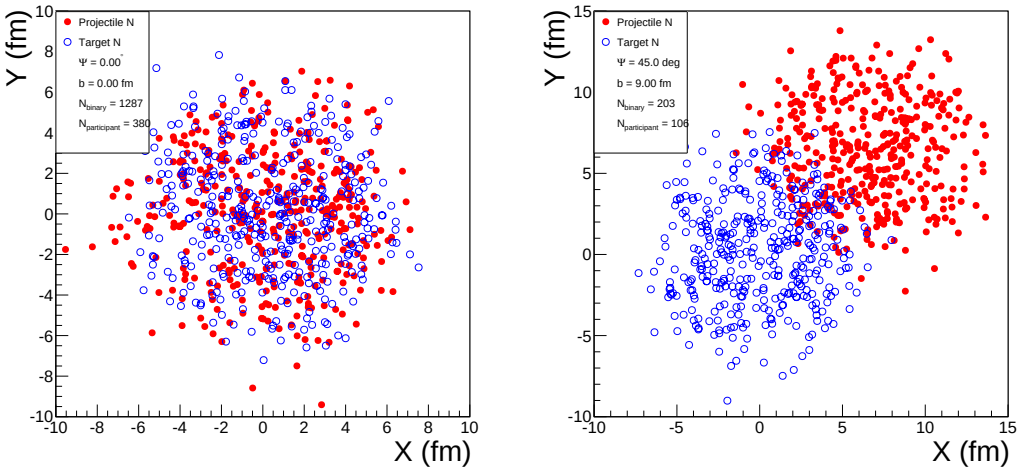
In relativistic nuclear collision experiments, the nuclei are accelerated to speed close to speed of light. Two approaching nuclei are confined to travel in straight line and are made to collide at the center of detector that measures the distribution of particles produced right after the collision. For convenience, the nuclei are assumed to move along z-direction right before the collision happens. In such high speed, due to relativistic length contraction, the diameter along z-direction shrinks and nuclei become thin circular disc. Therefore, the position of each nucleon in each nucleus is given in terms of the x and y coordinates only.

To simulate the collision, without loss of generality, we can assume that target nucleus (center of target nucleus), T, is fixed at the origin, $\vec{P}(x, y, 0) = (0, 0, 0)$. The position of projectile (center of projectile nucleus), $\vec{P}(x^{'}, y^{'}, 0) = (b\ cos\Psi, b\ sin\Psi, 0)$, where the value of $\Psi \in [0, 2\pi]$ is chosen randomly such that the projectile collides the target in any quartile of transverse plane passing through the transverse plane of the target and b is the radial distance between centers of the target and projectile nuclei. Thus, during collision, for any nucleon in T and P:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\ sin\theta\ cos\phi \\ r\ sin\theta\ sin\phi \\ 0 \end{pmatrix} and \tag{2.5}$$

$$\begin{pmatrix} x^{'} \\ y^{'} \\ z^{'} \end{pmatrix} = \begin{pmatrix} r\ sin\theta\ cos\phi + b\ cos\psi \\ r\ sin\theta\ sin\phi + b\ sin\psi \\ 0 \end{pmatrix} \tag{2.6}$$

If d is the distance between any two nucleons from different nucleus P



(a) An event of MCG model for $Au+Au$ collision with impact parameter $b = 0fm$ and $\Psi = 0°$.

(b) An event of MCG model for $Au+Au$ collision with impact parameter $b = 9fm$ and $\Psi = 45°$.

Figure 2.4 Central and peripheral collision events

and T, the hadronic collision is assumed to take place if,

$$d \leq \sqrt{\frac{\sigma_{inel}^{nn}}{\pi}}, \tag{2.7}$$

where σ_{inel}^{nn} is the measured inelastic nucleon-nucleon cross section. It is collision energy dependent; for the collision energy, $\sqrt{s_{NN}} = 200GeV$, $\sigma_{inel}^{nn} \sim 42mb$.

During a collision event, the number of nucleons that meet the hadronic collision requirement, as in equation (2.7), are called participant nucleons N_{part} where as total number of binary pairs meeting the requirements are called N_{bin}. In Figure 2.5, the distribution of N_{part} and N_{bin} are shown in semi-log scale. It is seen that both distributions fall off almost exponentially in mid-multiplicity range.

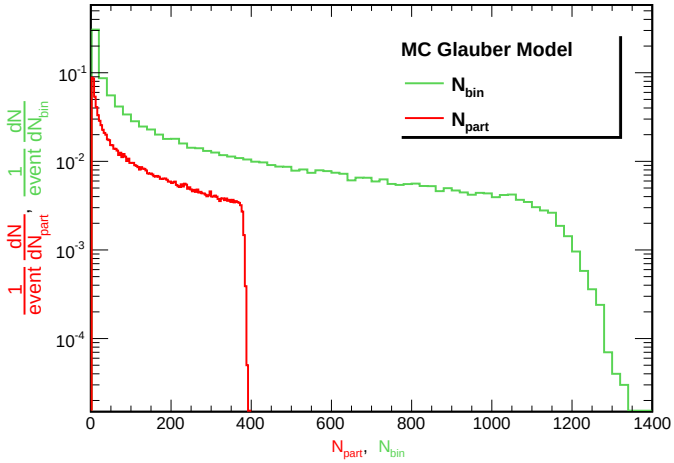


Figure 2.5 The N_{part} and N_{bin} distributions in MC Glauber Model.

The N_{part} and N_{bin} are decreasing functions of impact parameter b. Theoretical upper limit of N_{part} goes to sum of nucleons where as that of N_{bin} goes to the product of the nucleons in colliding nuclei.

The experimental set-up of the heavy-ion collision consists of two beams of ions approaching each other at speed very close to the speed of light and collide. A single collision is called an *event* and its precise location is called a *vertex*. As a consequence of the extremely high energy of collision, quark and gluons in the nuclei are expected to de-confine. The de-confined quarks and gluons shortly after collision re-combine to make stable particles such as pions, kaons, protons, electrons etc. At the same time these particles are thrust away from the collision center. In general, such an experiment consists of detectors centered around the point where the event occurs. With the aid of detector information, the path taken by the particles is traced and their direction along

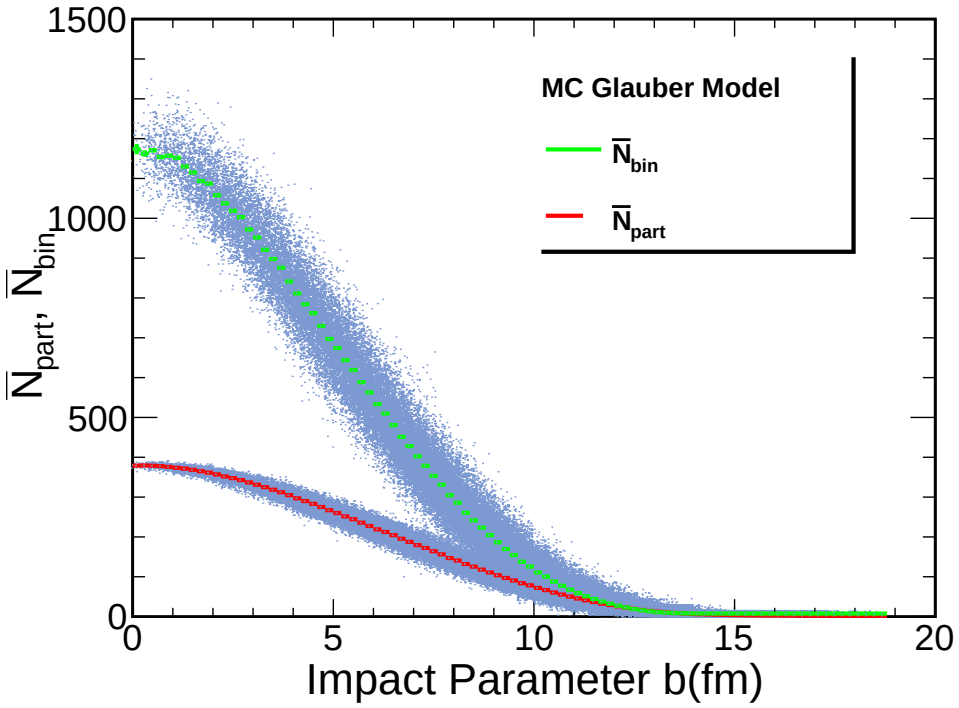


Figure 2.6 The fluctuation of N_{bin} and N_{part} for different values of b are shown in upper and lower bands respectively. The average profile of each count is given by the line in the middle of band.

with kinetic variables such as momentum is recorded. The path taken by a particle is called a *track*. The number of tracks produced is a function of the type of the colliding nuclei, the energy of collision and the nature of collision (peripheral or central collision). For $Au + Au$ collisions at 200GeV, an event produces particles in range of 2 to 2000. The Figure 2.7 shows schematic diagram of collision with variables used in the current study.

The side and front views of reconstructed tracks in a typical event in $Au + Au$ collision at $200GeV$ in STAR experiment is shown in Figure 2.8.

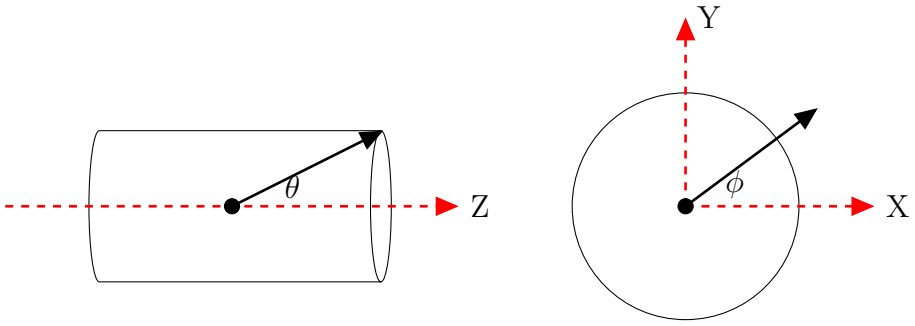


Figure 2.7 Cartoon of a track with its cylindrical coordinates. Left: lateral (side) view of a track starting from the collision vertex and making an angle θ with $z-axis$. The θ is related to $pseudo-rapidity(\eta)$ by relation : $\eta = -\left[ln(tan(\frac{\theta}{2}))\right]$. Right: the cross section (front) view of the track. It makes an angle of ϕ with $x-axis$ in $xy-plane$.

In the MCG model, particle production in heavy ion collision is described as a convolution of independent $p+p$ collisions [4]. The number of charged particles, n, produced in $p+p$ collisions at different energies have been measured and analyzed in different experiments. The charged particle multiplicity (n) in pp collisions is well described by a negative binomial distribution (NBD) [5]. The fitting parameters depend on pseudo rapidity (η) and collision energy(S).

$$P_{NB}(n;\bar{n},k) = \binom{n+k-1}{k-1} p^k \left(1-p\right)^n, \tag{2.8}$$

where $p = \left(1+\frac{\bar{n}}{k}\right)^{-1}$ is function of s and η. For $\sqrt{S_{pp}} = 200GeV$ and $|\eta| < 0.5$, the values of $\bar{n} = 2.48 \pm 0.06$ and $k = 2.0 \pm 0.2 \pm 0.1$ which gives $p = 0.446$ [5].

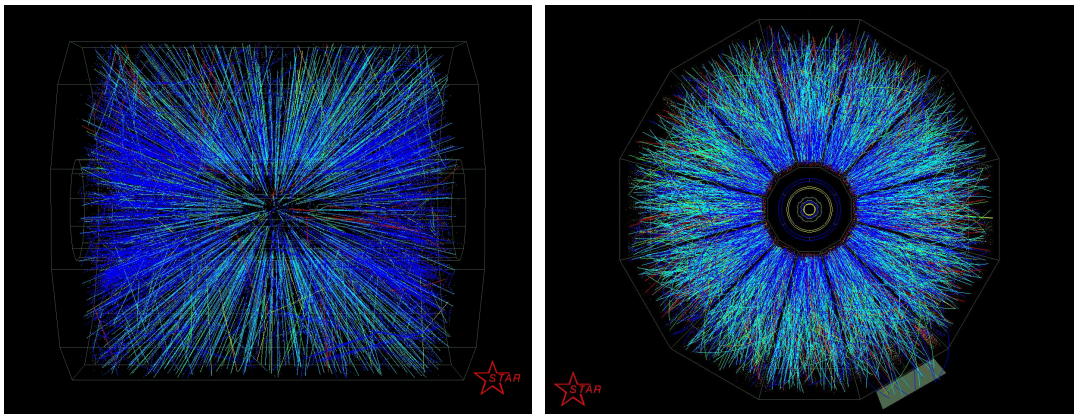


Figure 2.8 Reconstructed tracks in a typical event. Left: side view. Right: front view of the same event. The color scale represents the energy of the particle blue being low and red high.

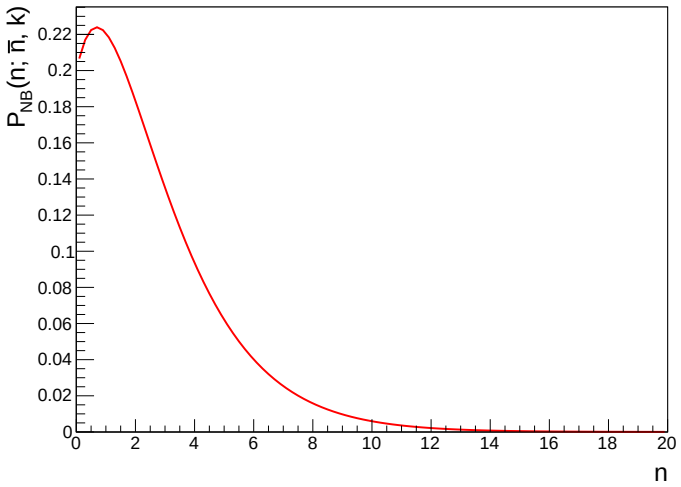


Figure 2.9 Charged particle multiplicity, n, in *pp* collision follows the trend of negative binomial distribution (NBD).

In high energy collision events, the multiplicity distribution (as in fig. 2.9) is the weighted superposition of two classes of events: hard and soft events. In hard collisions, the produced particle are sprayed creating one or more cone

like structures, called jets, where as in soft collisions, there is no such jet.

The average hard particle multiplicity yield is obtained from:

$$\bar{n}_{hard} = \frac{\frac{\alpha n^2}{\epsilon \Delta\eta}}{1 + \alpha n/(\epsilon \Delta\eta)}, \tag{2.9}$$

where $\alpha = 0.0105$, $\epsilon = 2.0$ and $\Delta\eta = 1.0$ [6].

The hard particle yield (n_{hard}) is obtained as a sample from the Poisson distribution with mean $\bar{n}_{hard}$.

The characterization of jet structure is beyond the scope of the current study. In heavy-ion collision, there are many hard interactions producing jets in every possible directions. Therefore, we assume that the hard particle production is oriented randomly in the azimuth.

Similarly, the soft particle yield is given by:

$$n_{soft} = \frac{n}{1 + \alpha n/(\epsilon \Delta\eta)}. \tag{2.10}$$

The direction of the soft particles (n_{soft}) is distributed such that there is quadrupole structure in the azimuthal direction:

$$\frac{dN}{d\phi} = \rho(\phi) \propto \left[1 + 2v_2 cos2(\phi - \psi)\right], \tag{2.11}$$

where ψ, called reaction plane angle, is azimuthal angle between colliding nucleons before collision happens. The v_2, related to the quadrupole amplitude (A_Q), gives the amplitude of azimuthal anisotropy. v_2 is described in detail in

the next section [5.1]. For a selected multiplicity, the A_Q is related to v_2 as shown below:

$$A_Q = \frac{n}{2\pi\Delta\eta} v_2^2. \tag{2.12}$$

The magnitude of A_Q is compared with quadrupole amplitude obtained from proton+proton experiment data fitting, $A_{Q,exp}$, to obtain the value of v_2.

$$A_{Q,exp} \approx \frac{n_{soft}}{n} \left[a_0 + a_1 \left(\frac{n_{soft}}{\Delta\eta} \right) + a_2 \left(\frac{n_{soft}}{\Delta\eta} \right)^2 \right], \tag{2.13}$$

where the fitting parameters, $a_0 = -0.000267$, $a_1 = 0.00048$ and $a_2 = 0.0000243$ [6].

In heavy ion collisions, a nucleon from the projectile nucleus is likely to collide (interact) with more than one nucleon in the target nucleus. The probability of soft-interaction producing soft particles $n_{soft} \sim 98.5\%$ where as the probability of semi-hard scattering producing hard particles $n_{hard} \sim 1.5\%$ at collision energy of 200GeV. Therefore, the entire soft particle production through soft interaction can be assumed to come from first nucleon-nucleon interaction for each nucleon in projectile nucleus. Whereas the hard particle production through hard interaction can be assumed to come from any of the rest of the nucleon-nucleon interactions. This way the particle production in nucleus-nucleus collision shows the two-component model with n_{soft} scaling with N_{part} and n_{hard} scaling with N_{bin} [6].

Fig.2.10 shows charge particle multiplicity for Au+Au minimum bias collisions at 200GeV in pseudo-rapidity $|\eta| < 0.5$ and 2π azimuth in log-log scale. It is seen that simulated result (dashed and dotted histogram) closely matches with real data (solid histogram). However, there is big discrepancy in low (left end of distribution) and high multiplicities (right of end of distribution). The detector efficiencies play important role to explain the discrepancy. Trigger and vertex finding inefficiencies are mostly responsible for discrepancy in the lower multiplicities where as trigger and particle trajectory reconstruction inefficiency are responsible for discrepancy in the higher multiplicities. General matching between simulated data and real data spectrum has been considered to be sufficient for the goal of present study.

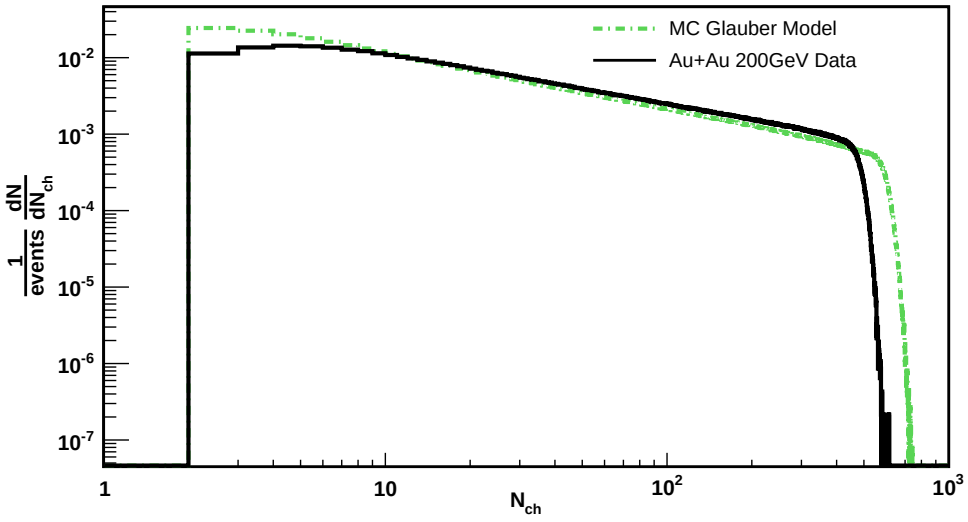


Figure 2.10 Total charge particle multiplicity distribution for Au+Au minimum bias collisions at 200GeV in pseudo-rapidity $|\eta| < 0.5$ and full azimuth normalized to one event.

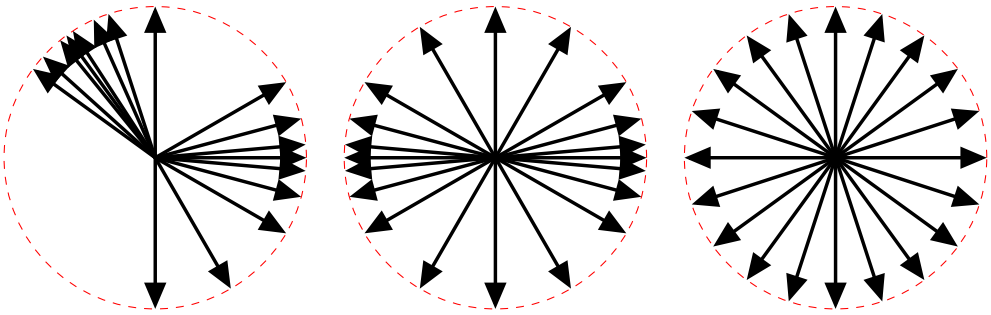


Figure 2.11 A toy example of particle distribution in transverse plane. Left: there are two jets of particles. Middle: tracks are distributed such that a quadrupole structure is formed by $\Delta\phi$, hence $v_2 > 0$. Right: tracks are uniformly distributed such that $v_2 = 0$.

Chapter 3

Correlation

The analysis of correlations provides essential information on the nature of the system that is produced in the aftermath of ultra-relativistic heavy-ion collisions. Quantum correlations, high-momentum jet correlations and correlations leading to collective behavior of system, called elliptic flow, have been the focus for the study of heavy-ion collisions. This section gives a general method for measuring two-particle correlations, where the two-particle pair obtained by combining all possible unique particle pairs produced in an event. The two-particle correlation measure is established using a standard definition; i.e., Pearson's correlation.

3.1 Two-particle correlation measure

In general, the number of particles produced in heavy-ion collisions are large enough that signal of interest is overwhelmed by the large background. The distribution of tracks in a typical event is shown in Figure 2.8. It it challenging

to extract the signals, if any, from such a vast background of particles. However, using correlations, the signals can be amplified to see otherwise obscured structures produced in the aftermath of a collision.

Figure. 2.11 shows three simplified hypothetical events with three distinct distributions of tracks in azimuthal direction. Figure. 2.11a shows an event with a two jets. Note that a *jet* is a spray of particles in a certain direction. Figure. 2.11b shows an event in which the tracks are distributed such that the number of tracks in opposite direction is large. Similarly, Figure. 2.11c shows an event in which all the tracks are randomly distributed giving no special structure. Note that the orientation of the jets or any structure is random in azimuth(ϕ) and in pseudo-rapidity (η).

Figure. 3.1 shows simplified hypothetical event wise distribution of the azimuthal angle ϕ. The bold solid line (red) [representing an event] is $cos(2\phi)$ distribution with some *Gaussian* noise on the top of random background. The dotted line (blue) [another event] is phase shifted to the solid line by $\pi/4$. The mean bin content in each bin is indicated by dashed straight line. The bin to bin fluctuation $n_a - \bar{n}_a$ and $n_b - \bar{n}_b$ are correlated where as $n_a - \bar{n}_a$ and $n_c - \bar{n}_c$ are anti-correlated.

The data system contains the event-to-event distribution of observables like momentum (p), azimuth (ϕ) and pseudo-rapidity (η) for the number of particles produced in the given event. The schematic representation of event-to-event distribution of an observable, x (η or ϕ), is shown in first panel of Fig. 3.2. The second panel represents the mean of the event-to-event mean

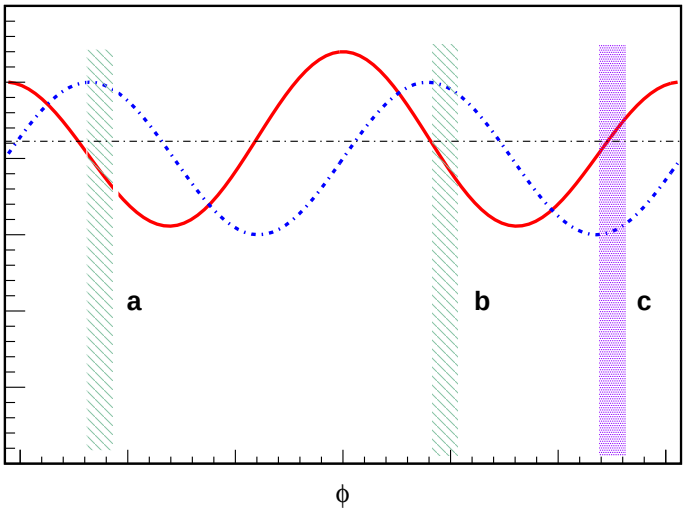


Figure 3.1 A simplified example of hypothetical distribution of phi for two events (solid red and dotted blue). The black straight line shows the average for all events. The fluctuations in bin content $(n_i - \bar{n}_i)$about mean at a and b are correlated and they are anti-correlated at a and c.

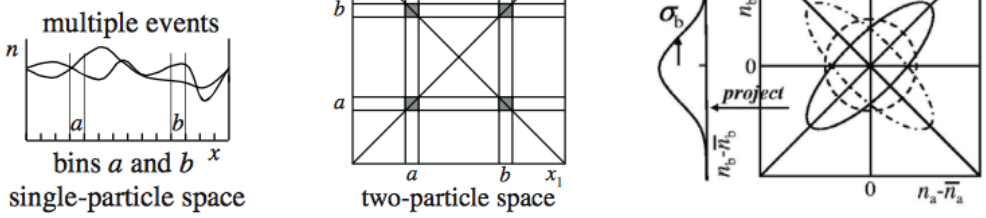


Figure 3.2 Event-to-event distribution of an observable (left). Two dimensional distribution of the observable with selected bin contents in bin a and b high-lighted (middle). Different possibilities of distributions between $(n_a - \bar{n}_a)$ and $(n_b - \bar{n}_b)$ (right) [1].

of the distributions in 2D space (x_1, x_2). Different possibilities of frequency distributions for $(n_a - \bar{n}_a)$ and $(n_b - \bar{n}_b)$, where a and b are any two bins, are

shown in the third panel. The ellipse along the axis labelled Σ indicates the correlation, along the axis labelled Δ indicates the anti-correlation and the circle indicates uncorrelated randomness between the bins [1].

The correlations between the contents of two arbitrary bins a and b are given using definition of Pearson's correlation coefficient.

$$\begin{aligned} Corr(a,b) &= \frac{Cov(a,b)}{\sqrt{Var(a).Var(b)}} \\ &= \frac{\frac{1}{N}\sum_{i=1}^{N}\left(n_i-\bar{n}\right)_a\left(n_i-\bar{n}\right)_b}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(n_i-\bar{n})_a^2 \;\; \frac{1}{N}\sum_{i=1}^{N}(n_i-\bar{n})_b^2}} \\ &= \frac{\overline{(n-\bar{n})_a(n-\bar{n})_b}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(n_i-\bar{n})_a^2 \;\; \frac{1}{N}\sum_{i=1}^{N}(n_i-\bar{n})_b^2}}, \end{aligned} \tag{3.1}$$

where N represents the number of events and $\bar{n}$ represents the average bin content over the events for the given bin.

In the context of heavy-ion collision, the contents of a bin are approximately distributed following Poisson distribution, where mean and variance are equal [i.e., $\frac{1}{N}\sum_{i=1}^{N}(n_i-\bar{n})_a^2 = \bar{n}_a$] [1]. Using the Poisson approximation, equation (3.1) can simplified to:

$$
\begin{aligned}
Corr(a,b) &= \frac{\overline{n_a n_b} - \bar{n}_a \bar{n}_b}{\sqrt{\bar{n}_a \bar{n}_b}} \\
&= \frac{\overline{n_a n_b/\epsilon} - \bar{n}_a \bar{n}_b/\epsilon}{\sqrt{\bar{n}_a \bar{n}_b/\epsilon^2}} \\
&= \frac{\Delta\rho}{\sqrt{\rho_{ref}}},
\end{aligned}
\tag{3.2}
$$

where ϵ is product of bin widths for bin a and b. Introduction of ϵ removes the dependencies on bin width of histogram. The quantity $\frac{\Delta\rho}{\sqrt{\rho_{ref}}}$ is interpreted as the number of correlated pairs per final-state particle.

Our objective is to extract any correlation signal coming from the particle tracks from an event. We define the sibling density, $\rho_{sib} = \overline{n_a n_b/\epsilon}$ such that only the tracks from the same event are paired. The density ρ_{sib} contains a large background coming from the uncorrelated pairs. Similarly, we define the reference density, $\rho_{ref} = \bar{n}_a \bar{n}_b/\epsilon$ such that the tracks from different events are paired. The track pairs from different events are similar to sibling pairs but they are not correlated. Thus, the difference in densities, $\Delta\rho = \rho_{sib} - \rho_{ref}$ is expected to be pure signal. However, as an experimental artifact such as detector acceptance and inefficiencies, both the sibling or reference distribution contain some structure. Using the ratio ρ_{sib}/ρ_{ref}, such structure coming from detector artifacts can be removed. The equation (3.2) is rewritten as:

$$
\begin{aligned}
Corr(a,b) &= \frac{\Delta\rho}{\sqrt{\rho_{ref}}} = \frac{\rho_{sib} - \rho_{ref}}{\sqrt{\rho_{ref}}} \\
&= \sqrt{\rho_{ref}}(\frac{\rho_{sib} - \rho_{ref}}{\rho_{ref}}) \\
&= \sqrt{\rho_{ref}}(r-1),
\end{aligned}
\tag{3.3}
$$

where $r = \rho_{sib}/\rho_{mix}$. Using the efficiency corrected pre-factor $\sqrt{\rho_{ref}}$ the final expression of correlation is written as:

$$
Corr(a,b) = \sqrt{\rho'_{ref}}(r-1),
\tag{3.4}
$$

where the ($'$) in $\sqrt{\rho'_{ref}}$ indicates that the density is efficiency corrected. It is a constant for the present analysis.

Angular correlations in difference variables such as $\Delta\eta = \eta_1 - \eta_2$ and $\Delta\phi = \phi_1 - \phi 2$ are typical in $Au - Au$ collisions at RHIC [2]. In the context of the present article, the variables (ϕ_1, ϕ_2) are expressed in terms of sum and difference variables: $(\phi_1, \phi_2) \rightarrow (\phi_1 + \phi_2, \phi_1 - \phi_2) = (\Sigma\phi, \Delta\phi)$, where $\Sigma\phi = \phi_1 + \phi_2$ and $\Delta\phi = \phi_1 - \phi_2$. The data distribution on $\Sigma\phi$ is stationary and is independent of the sum variable $\Sigma\phi$ [1]. In the context of heavy-ion collisions, the orientation of the collision (i.e., ψ in Fig. 2.4) is random, which in turn causes randomness in the absolute position of ϕ. In such distributions the mean and variance do not depend on absolute position and is referred

to as a stationary distribution. For such a stationary distribution, the projection along the difference axis contains all the relevant information of the distribution. That is $(\phi_1, \phi_2) \rightarrow (\Delta\phi)$. In practice, ρ_{sib} is obtained from the distribution of $(\phi_1 - \phi_2)$, where both particles coming from the same event whereas ρ_{ref} is obtained from the distribution of $(\phi_1 - \phi_2)$ each particle from different events.

Chapter 4

Results

The results presented in this section are not normalized to compare the results with experimental data. There is a constant factor to multiply the correlation to compare with the results from experimental data.

The two particle azimuthal correlation was calculated for different sets of data. The effect of number of events on the magnitude of the correlation amplitude, v_2, was studied. For the number of events up to a million, as seen in Figure 4.1, the correlation amplitude is not visually and statistically significant. However, as we increase the number of event samples to 4 and 10 millions of events, the correlation becomes successively significant. The correlations from 4 and 10 million events are presented in Figures 4.2 and 4.3 respectively. The result suggests that a significant v_2 is obtained from a sample of about 10 million simulated events only. In the real data, however, about a million events produces a statistically significant v_2.

In the present simulation, using v_2 from $p + p$ collisions as an input,

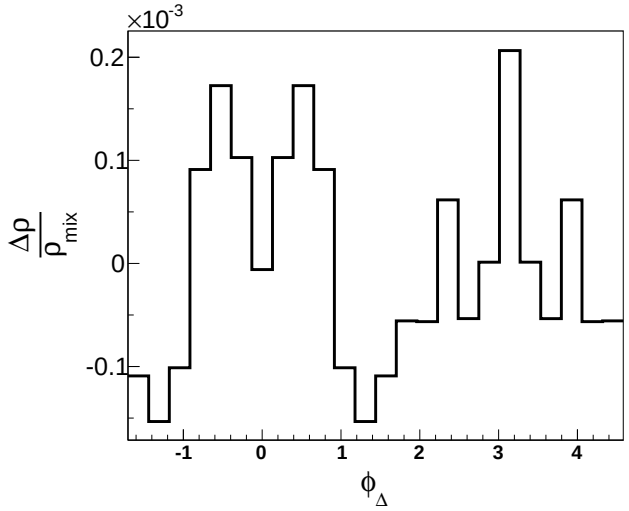


Figure 4.1 Azimuthal correlations for a $\sim$ million events.

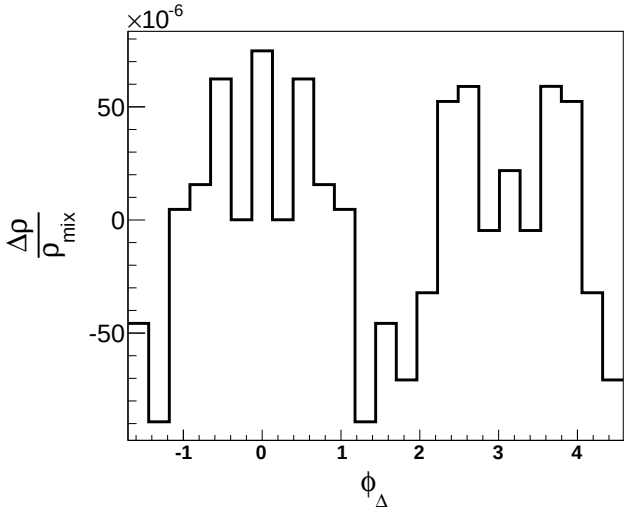


Figure 4.2 Azimuthal correlations for ~ 4 million events.

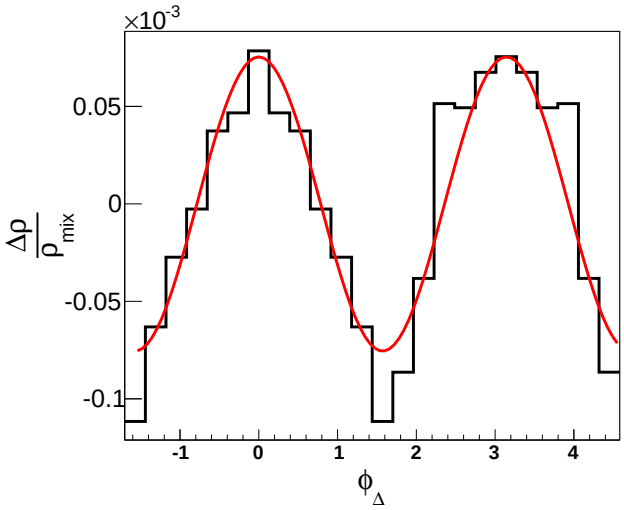


Figure 4.3 Azimuthal correlations for ~ 10 million events.

we want to estimate the v_2 in $Au + Au$ collisions. As a test of the simulation, the effect of magnitude of v_2 from $p + p$ collisions have been studied. In the Figures 4.4 and 4.5 the correlations of a million events using 2 and 4 times the measured v_2 from $p + p$ collisions is presented. The magnitude of v_2 in $Au + Au$ collisions is sensitive to magnitude of v_2 in $p + p$ collision; four fold increase in source leads to about 15 times increase in result.

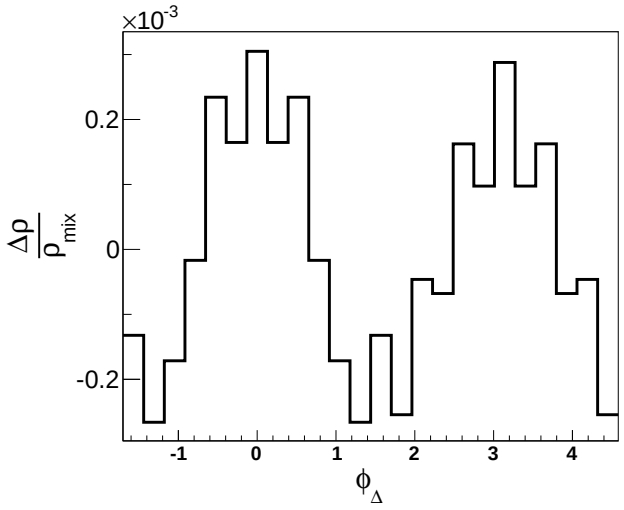


Figure 4.4 Azimuthal correlations for ∼million events. The magnitude of v_2 used was twice the measured v_2 in $p + p$ collisions.

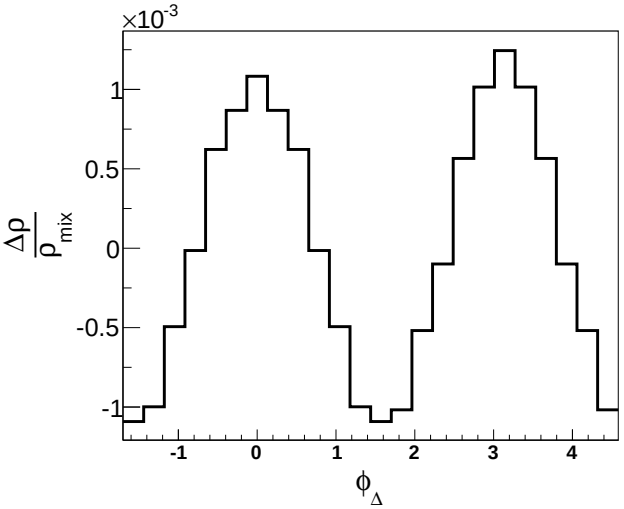


Figure 4.5 Azimuthal correlations for ∼million events. The magnitude of v_2 used was four times the measured v_2 in $p + p$ collisions.

Because of the geometry of overlap during the collision, the magnitude of v_2 is expected to be highest in mid-central collisions (40-60% centrality). The correlations were measured for the mid-central collisions ($b = 10$fm). In Figure 4.6 correlations for 10 million events for mid-central collisions are shown. The error bars shown are statistical errors.

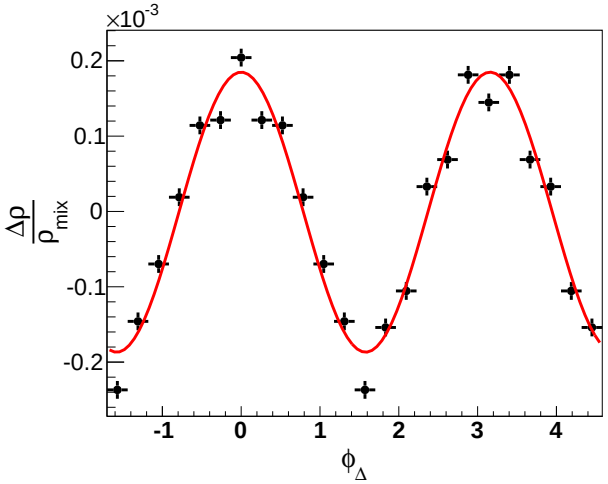


Figure 4.6 Azimuthal correlations for $\sim$ 10million events. The value of b was fixed for mid-central collision (40-60% centrality, $b = 10fm$). Compared to Fig. 4.3, amplitude of the correlation has increased by three fold. In mid-central collision, the overlap between the nuclei is almond shaped giving rise to larger v_2.

Chapter 5

Discussion

5.1 Convolution of Quadrupole

In high energy heavy-ion nuclear collisions, event-wise particle density on azimuth ($\frac{dN}{d\phi}$) is anisotropic and is represented in 1D Fourier expansion:

$$\frac{dN}{d\phi} = \rho(\phi) \propto \left[1 + 2\sum_{n=1}^{\infty} v_n cos[n(\phi - \Psi)]\right], \tag{5.1}$$

where Ψ, called reaction plane angle, is the angle between centers of nuclei in the transverse plane where the collision happens and x-axis of the transverse plane. The v_n are the Fourier coefficients which have been subject of study in heavy-ion community for two decades. Although, the physical interpretations of the azimuthal anisotropy varies among scholars, the Fourier coefficients have been measured in good precision [7][8]. Among the coefficients, the second coefficient of the Fourier expansion, v_2 has largest measured magnitude and it is subject of interest of current study.

According to a school of thought, high energy heavy-ion collision produces de-confined state of quarks and gluons, called Quark Gluon Plasma (QGP). This plasma soup behaves as a fluid and it flows giving rise to the anisotropy depending on the geometry of collision. In particular, in this school of thought, v_2, also called *elliptic flow*, is consequence of flow of classical fluid [8]. Other school interprets the anisotropy as a manifestation of gluonic multipole radiation [7].

In $p+p$ collision (i.e. nucleon-nucleon collision), because of small size of the proton, the Quark and Gluon Plasma fluid is not supposed to be formed and hence *elliptic flow* is expected to be zero. However, contradicting to the expectation of the flow interpretation, experimental $p+p$ collision data doesn't rule out the presence of azimuthal anisotropy. In Ref. [6], the amplitude of azimuthal quadrupole correlation (A_Q) in $p+p$ 200GeV and high multiplicity ($N_{trk}^{offline} \geq 110$) 7TeV data is reported as 0.00135±0.00009 and 0.0147±0.0029 respectively. Note that the (A_Q) is related to v_2 as shown in equation (2.12).

A heavy-ion collision (eg. Gold+Gold collision) can be considered as a superposition of multiple $p+p$ collisions in a classical picture of mechanics. The non-zero v_2 in $p+p$ collisions, therefore, is likely to add together to give cumulative v_2 in heavy-ion collisions. This is the main motivation of the current simulation.

Equation (2.10) gives the number of particles produced, n_{soft}, in soft interaction between two nucleons. These particles are distributed in azimuthal space such that we get measure magnitude of azimuthal quadrupole correla-

tions (A_Q). Many such collisions are collected together to create a heavy-ion event. Finally, using Pearson's correlation coefficient, the correlation amplitude is calculated for many heavy-ion events in a way described above.

In Chapter 4 Figure 4.6, the azimuthal correlations from simulated gold gold collision events have been shown. In the figure, it is clearly seen that the correlation magnitude is statistically significant. To compare directly this result to the correlation results from real experimental data, it must be multiplied by constant multiplicative factor, $\frac{d^2N}{d\eta d\phi}$ [2]. The v_2 value extracted from Figure 4.6 is smaller compared to v_2 measured in experimental data. However, looking at the result, we can't rule out a possibility that a significant fraction of v_2 in heavy-ion collision is contributed by v_2 from nucleon+nucleon collisions. Further investigation with this model could give us better understanding of v_2 observed in heavy-ion collisions.

Appendix

Appendix 1

Appendix

The maind code for the simulation consists of the following files.

```
/*
Written by Prabhat Bhattarai
Date: Dec 2015
Language C++ and Object oriented Program ROOT.

Monte Carlo Glauber Simulation for Heavy Ion Collisions.
This model calculates Nch, Npart, Nbinary

Most of the references are referenced here:
J. Phys.G: Nucl. Part.Phys. 35(2008) 125106(22pp)
An Article by Lanny Ray and M. Daugherity

To run:
compile using g++. Note that you need root installed in your node.
g++ -o executable MCGSimulation.C `root-config --cflags --glibs`
To execute:
./executable
*/

#include "TMath.h"
#include "MCGSimulation.h"
#include <iostream>
```

```
#include "TROOT.h"
#include "TH1.h"
#include "TH2.h"
#include "TH3.h"
#include "TRandom.h"
#include "TPolyLine3D.h"
#include "TTimeStamp.h"
#include "TLegend.h"
#include "TLegendEntry.h"
#include "TF1.h"
#include "TCanvas.h"
#include "TApplication.h"
#include <fstream>
#include "TStyle.h"
using namespace std;
double RadialPos_of_Nucleons();
double NchtoCentrality(int Nch);
double Get_AQ(int Centrality);
double Get_V2(int Nch, double Delta_eta, double AQ);
double Get_AQ_using_nsi_nchi(int nch_i, int ns_i, double DeltaEta);
double Get_phi_ij_from_dN_over_dPhi_Dist(ofstream &outfile,
       int event,int Nch, double V2, double psi);
double Get_Random_phi_ij(ofstream &outfile, int event, int nh_ij);
double NbarChofb(double npp, double Kharjeev_x, double Npart,
    double Nbin);
double Negative_Binomial_pdf(double nbar, unsigned int k);
double LannyDist(double r1,double theta1,double phi1,double r2,
       double theta2,double phi2,double b);
double GetDistAtoB(double r1,double theta1,double phi1,double r2,
       double theta2,double phi2,double b,double psi);
double GetsigmaNN(double b,double sigma0,double beta,double b0,
    double zb);
const double pi = TMath::Pi();
const double twopi = 2.0*pi;


int main(){
```

```
//In Polar Coordinate Positions of A and B Nucleons
double PosofNucleon_A[NoofNucleons_A+2];
double PosofNucleon_B[NoofNucleons_B+2];
double PhiofNucleon_A[NoofNucleons_A+2];
double PhiofNucleon_B[NoofNucleons_B+2];
double ThetaofNucleon_A[NoofNucleons_A+2];
double ThetaofNucleon_B[NoofNucleons_B+2];

// In Cartesian Coordinate Position of A Nucleons
double xcor_A[NoofNucleons_A+2];
double ycor_A[NoofNucleons_A+2];
double zcor_A[NoofNucleons_A+2];

// In cartesian coordinate the max value of z for
int   Max_z_cor_A_NucleonID;

//In cartesian coordinate. Position of B_Nucleons
double xcor_B[NoofNucleons_B+2];
double ycor_B[NoofNucleons_B+2];
double zcor_B[NoofNucleons_B+2];

// Azimuthal orientation of Nulceons in A and Nucleons in B
double NucleonPair_Psi[NoofNucleons_A+2][NoofNucleons_B+2];
//Output canvas
TCanvas *can = new TCanvas("canvas", "canvas",2000,800);
can->Divide(4,3);

//Time stamp to write start time of program
TTimeStamp TotSstart;
int startTime = TotSstart.GetSec();

const double pi = TMath::Pi();
const double piover4 = TMath::Pi()/4.0;


//Create Nch Histogram;
```

```
//TH1D *PosProfile_A = new TH1D("r_A","r_A",100,0,10);
//TH1D *ThetaProfile_A = new TH1D("Theta_A","Theta_A",100,0,pi);
//TH1D *PhiProfile_A = new TH1D("Phi_A","Phi_A",100,0,twopi);
//TH3D *NucleusA = new TH3D("Nucleons_Pos_A","Nucleon_Pos_A
    ",100,-10,15,100,-10,15,100,-10,15);
//TH3D *NucleusB = new TH3D("Nucleons_Pos_B","Nucleon_Pos_B
    ",100,-10,15,100,-10,15,100,-10,15);
TH3D *NucleusA = new TH3D("Nucleons_Pos_A","Nucleon_Pos_A"
    ,100,-10,10,100,-10,10,100,-10,10);
TH3D *NucleusB = new TH3D("Nucleons_Pos_B","Nucleon_Pos_B"
    ,100,-10,10,100,-10,10,100,-10,10);
TH1D *PosProfile_B = new TH1D("r_B","r_B",100,0,10);
TH1D *ThetaProfile_B = new TH1D("Theta_B","Theta_B",100,0,pi);
TH1D *PhiProfile_B = new TH1D("Phi_B","Phi_B",100,0,twopi);
TH1D *PairWisePsihist = new TH1D("PairWisePsi","PairWisePsihist"
    ,100,0,twopi);


TH1D *bhist = new TH1D("ImpactParm","ImpactParmeter b",100,0,bmax
    );
TH1D *psihist = new TH1D ("Psi","#Psi ",100,0,twopi);
TH1D *Nbinhist= new TH1D("Nbinevent","Nbinevent",100,0,2000);
TH1D *Nparthist= new TH1D("Npartevent","Npartevent",100,0,400);
TH1D *Nchhist= new TH1D("Nchevent","Nchevent",100,0,2000);
TH1D *Nch14hist= new TH1D("Nch14event","Nch14event",100,0,7);
TH1D *Npart14hist= new TH1D("Npart14event","Npart14event",100,0,7);
TH1D *Nbin16hist= new TH1D("Nbin16event","Nbin16event",100,0,7);

TH1D *DistanceAtoB= new TH1D("DistanceAtoB","DistanceAtoB"
    ,100,0,50);
TH1D *L_DistanceAtoB= new TH1D("L_DistanceAtoB","L_DistanceAtoB"
    ,100,0,50);

//TH1D * NegBinomial_nch_i = new TH1D("NegBinomial_nch_i","
    NegBinomial_nch_i",20,0,20);
TH1D * SoftperNucleus_ns_i = new TH1D("SoftperNucleus_ns_i","
    SoftperNucleus_ns_i",1000,0,1000);
```

```
TH1D * HardperNucleus_ns_i = new TH1D("HardperNucleus_ns_i","
    HardperNucleus_ns_i",1000,0,1000);
TH1D * SoftPlusHardperNucleus_ns_ij = new TH1D("
    SoftPlusHardperNucleus_ns_ij",
          "SoftPlusHardperNucleus_ns_ij",1000,0,1000);
TH1D * SoftPlusHard14 = new TH1D("SoftPlusHard14","SoftPlusHard14"
    ,100,0,7);
TH2D *bVersusNpart = new TH2D("bVersusNpart","bVersusNpart"
    ,100,0,20,100,0,400);
TH2D *bVersusNBin = new TH2D("bVersusNBin","bVersusNBin"
    ,100,0,20,100,0,1500);

//For Radius of Nucleus;
//TF1 *f1 = new TF1("f1","x*x*1.0/(1.0+exp((x-6.43)/0.568))",0.0,10.0);

//For dNoverdPhi
TH1D *RandomPhihist= new TH1D("RandomPhihist","RandomPhihist"
    ,100,0,twopi);

//Create Random number
TRandom random(0);//Randon Number Generator
TTimeStamp TS;
int startNanoTime = TS.GetNanoSec();
int startDate= TS.GetDate();
int seed = (startDate+startNanoTime );
//This seed is based on Nanosecond and Date. So I hope it will
//help me to produce real random numbers.
//random.SetSeed((unsigned int)time(NULL)+seed);



//EventID and Track _ij Values in output file
ofstream outfile;
outfile.open("EventIDTrackPhi.txt");

sigmaNNofb =sigma_inel_0;// GetsigmaNN(b,sigma_inel_0,beta,b0,zbthick);
sigmabCut= sigmaNNofb/TMath::Pi();
```

```
//I am not doing suare root of this term. It saves my time while
    comparing with square of distance.

//Create Event Loop
event=1;
do {


//
    ***************************************************************************

//Random distribution of Nucleons of Nucleus A
//
    ***************************************************************************

for(int i=1;i<=NoofNucleons_A;i++){
  PosofNucleon_A[i] =RadialPos_of_Nucleons();
  PhiofNucleon_A[i] = 2.0*pi*random.Rndm();
  ThetaofNucleon_A[i]= acos(1.0-2.0*random.Rndm());
  //From geometry shoudn't be uniform dist.

  //  PosProfile_A->Fill(PosofNucleon_A[i]);
  //  ThetaProfile_A->Fill(ThetaofNucleon_A[i]);
  //  PhiProfile_A->Fill(PhiofNucleon_A[i]);

  //In Cartesian Coordinate System
  xcor_A[i]= PosofNucleon_A[i]*sin(ThetaofNucleon_A[i])*cos(
      PhiofNucleon_A[i]);
  ycor_A[i]= PosofNucleon_A[i]*sin(ThetaofNucleon_A[i])*sin(
      PhiofNucleon_A[i]);
  zcor_A[i] =PosofNucleon_A[i]*cos(ThetaofNucleon_A[i]);
  NucleusA->Fill(xcor_A[i],ycor_A[i],zcor_A[i]) ;
}
//
    ***************************************************************************
```

```
//
    ************************************************************************

// Random distribution of Nucleons of Nucleus B
//
    ************************************************************************


//The probability of landing center of projectile nucleus is proportional
//to area 2*pi*b*db. Note that it is two dimensional case. There is
//pretty narrow chance that random number hits center.
//I plan to fit the distribution of b with linear function 2*pi*xdx
//I can generate linear random variable
// TF1 *f1_b = new TF1("f1_b","x",0,bmax);
// b= f1_b->GetRandom();
// delete f1_b;   //Free memory by deleting f1_b;

//We can aslo choose to do the following


// b= 0.0;  //If wanted fixed b
b=bmax*sqrt(random.Rndm()); //Why? X transformation of Uniform dist
    is linear.

//Generate a random angle along phi direction. Collision
// happens randomly at an angle psi from X axis
//   psi=0;
//   psi =0.0;
//   psi=piover4; //If wanted fixed Psi
psi=twopi*random.Rndm(); //Psi angle covers entire phi (azimuthal) space

for(int i=1;i<=NoofNucleons_B;i++){
  PosofNucleon_B[i] =RadialPos_of_Nucleons();
  PhiofNucleon_B[i] = 2.0*pi*random.Rndm();
  ThetaofNucleon_B[i]= acos(1.0-2.0*random.Rndm());
```

```
  PosProfile_B->Fill(PosofNucleon_B[i]);
  ThetaProfile_B->Fill(ThetaofNucleon_B[i]);
  PhiProfile_B->Fill(PhiofNucleon_B[i]);

  //In Cartesian Coordinates
  xcor_B[i]= PosofNucleon_B[i]*sin(ThetaofNucleon_B[i])*cos(
      PhiofNucleon_B[i])+b*cos(psi);
  ycor_B[i]= PosofNucleon_B[i]*sin(ThetaofNucleon_B[i])*sin(
      PhiofNucleon_B[i])+b*sin(psi);
  zcor_B[i] =PosofNucleon_B[i]*cos(ThetaofNucleon_B[i]);
  NucleusB->Fill(xcor_B[i],ycor_B[i],zcor_B[i]) ;
}

//
    ************************************************************************




//
    ************************************************************************

// Stablish Collision condition and find Nbinevent. Prepare to find
    Npartevent
//
    ************************************************************************


//Collision condition requires that:
//DistAtoB <   (sigmaNNofb/)= sigmabCut
//sigmaNNofb is defined in function GetsigmaNN
//  sigmaNNofb =sigma_inel_0;// GetsigmaNN(b,sigma_inel_0,beta,b0,
    zbthick);
//sigmabCut= sqrt(sigmaNNofb/TMath::Pi());
//Check for faster algorithm
//   sigmabCut= sigmaNNofb/TMath::Pi();
```

```
// Initialization  of  Nbinary, Nspectator, and Nparticipants
Nspecevent=0;
Npartevent=0;
Nbinevent=0;

Awounded[0]=0;
Bwounded[0]=0;
TH2D *NpartCounter = new TH2D("NpartCounter","NpartCounter",
    NoofNucleons_A,1,
          NoofNucleons_A,NoofNucleons_B,1,NoofNucleons_A);

//For Nucleus A
for (int NucA=1;NucA<=NoofNucleons_A;NucA++){
//For Nucleus B
for (int NucB=1;NucB<=NoofNucleons_B;NucB++){

double x1,y1,x2,y2;
x1 = PosofNucleon_A[NucA]*sin(ThetaofNucleon_A[NucA])*cos(
    PhiofNucleon_A[NucA]);
y1 = PosofNucleon_A[NucA]*sin(ThetaofNucleon_A[NucA])*sin(
    PhiofNucleon_A[NucA]);

x2= PosofNucleon_B[NucB]*sin(ThetaofNucleon_B[NucB])*cos(
    PhiofNucleon_B[NucB])+b*cos(psi);
//Offset  is  along any direction  in  XY plane
y2= PosofNucleon_B[NucB]*sin(ThetaofNucleon_B[NucB])*sin(
    PhiofNucleon_B[NucB])+b*sin(psi);
//Offset  is  along any direction  in  XY plane

double abs_xdistance =TMath::Abs(x1-x2);
double abs_ydistance =TMath::Abs(y1-y2);
//Calculation of distance  A to B takes longer  time.  So it  is  optimized by
//absolute distance.
/*DistAtoB= GetDistAtoB(PosofNucleon_A[NucA],ThetaofNucleon_A[NucA
    ],
  PhiofNucleon_A[NucA],PosofNucleon_B[NucB],ThetaofNucleon_B[NucB],
```

```
  PhiofNucleon_B[NucB],b, psi);
 */
if(abs_xdistance<=sigmabCut && abs_ydistance<=sigmabCut){
  //I dont want to calculate distance if A and B are too far.
  //If both of the absolute distances are less than the sigmaCut, the
  //distance has to be less than the SigmaCut as well.

  DistAtoB = abs_xdistance*abs_xdistance+ abs_ydistance*abs_ydistance;
  //Square root takes more time so, I am not doing square root. The
  //square has to work as well provided I squared sigmaCut value.


  DistanceAtoB->Fill(DistAtoB);
  if (DistAtoB<sigmabCut){
    Awounded[NucA]=1;
    Bwounded[NucB]=1;

    Nbinevent= Nbinevent+1;
    NpartCounter->SetBinContent(NucA,NucB,Bwounded[NucB]);
  }
}
}//else
}

//
    **************************************************************************

//Is each binary collision is tagged?
//  Check!! Yes , each participant is tagged
//
    **************************************************************************


// Particlie production is supposed to be due to hard and soft processes.
// In general hard refers to higer energy jet like sturctures where as
// soft means low energy
```

```
double nh_i_From_nch_ij=0;
//hard multiplicity in each interaction of nucleons i and j
int Tot_soft_ns_i =0;//Total soft multiplicity
int Tot_hard_ns_i=0;//Total soft multiplicity
int SoftPlusHardTrkID= 0;//Total soft plus hard multiplicity

//
    ************************************************************************

//For Maximum Z position
//Currently I have implemented random z position, that comes first in the
//loop.
//
    ************************************************************************

//Max_z_cor_A=-10.0; //Lowest possible value
Max_z_cor_A_NucleonID= 0;// A nucleon ID corresponding to Max_z_cor_A

//For Nucleus B .B is projectile Nucleus with NoofNucleons_B nucleons.
for (int NucB=1;NucB<=NoofNucleons_B;NucB++){
  //For Nucleus A . It is target with NoofNucleons_A nucleons
  int Tot_nh_i_From_nch_ij=0;//Total hard multiplicity in interaction of
      nucleons i and j
  double * Max_z_cor_A = new double[NoofNucleons_A];
  //Maximum z of all participant in A for a projectile nucleon in B

  int NumPart_inA_perPart_inB=0;
  Max_z_cor_A[NumPart_inA_perPart_inB]=-10.0;

  for (int NucA=1;NucA<=NoofNucleons_A;NucA++){

if(NpartCounter->GetBinContent(NucA,NucB)==1){
  //For each NucB, if will find all NucA that gets wounded.

  //In Cartesian Coordinate System
  xcor_A[NucA]= PosofNucleon_A[NucA]*sin(ThetaofNucleon_A[NucA])*cos
```

```
    (PhiofNucleon_A[NucA]);
ycor_A[NucA]= PosofNucleon_A[NucA]*sin(ThetaofNucleon_A[NucA])*sin
    (PhiofNucleon_A[NucA]);
zcor_A[NucA] =PosofNucleon_A[NucA]*cos(ThetaofNucleon_A[NucA]);

xcor_B[NucB]= PosofNucleon_B[NucB]*sin(ThetaofNucleon_B[NucB])
   *cos(PhiofNucleon_B[NucB])+b*cos(psi);
ycor_B[NucB]= PosofNucleon_B[NucB]*sin(ThetaofNucleon_B[NucB])
   *sin(PhiofNucleon_B[NucB])+b*sin(psi);
zcor_B[NucB] =PosofNucleon_B[NucB]*cos(ThetaofNucleon_B[NucB]);

//This gives the nucleon in Nucleus A that is hit first by nucleons in
//Nucleus B
if (zcor_A[NucA]>= Max_z_cor_A[NumPart_inA_perPart_inB]){
  Max_z_cor_A[NumPart_inA_perPart_inB]= zcor_A[NucA];
  Max_z_cor_A_NucleonID= NucA;
}
else Max_z_cor_A[NumPart_inA_perPart_inB]=-10.0;


NumPart_inA_perPart_inB+=1;

//To find azimuthal orientation between each projectile and target
//nucleon from the reference point of nucleon in target B.
//Psi = arctan((y2-y1)/(x2-x1))
double DiffinY= ycor_B[NucB]-ycor_A[NucA];
double DiffinX= xcor_B[NucB]-xcor_A[NucA];
if(DiffinX!=0)NucleonPair_Psi[NucB][NucA]= atan(DiffinY/DiffinX);
if(DiffinX>=0&&DiffinY>=0)NucleonPair_Psi[NucB][NucA]+=0;
else if (DiffinX<0&&DiffinY>=0)NucleonPair_Psi[NucB][NucA]+=pi;
else if (DiffinX<0&&DiffinY<0)NucleonPair_Psi[NucB][NucA]+= pi;
else NucleonPair_Psi[NucB][NucA] += twopi;

//PNBD(NBinomial_n,NBinomial_nbar,NBinomial_k)
//NOTE: n is equivalent to x in
//the distribution used in Negative_Binomial_pdf() definitions
//Using Reference Z.Phys.C 43,357(1989)
```

```
int Random_BinaryColliding_nucleon; //Number should be between 1 and
    197.
double NBinomial_nbar= 2.48 ; //0.06 , for mod( )<=0.5 //Negative
    Binomial nbar
double NBinomial_k = 2; //0.1  //Negative Binomial K
double NBinomial_nch_i;
int nch_i;
double ns_i_From_nch_i; //Soft multiplicity
double Alpha_for_nsh_i= 0.0105; //Both for soft and hard
double Epsilon_for_nsh_i= 2.0;  //Both for soft and hard

//
    **********************************************************************

//Negative Binomial Distribution, each nucleon collision creates
//NBinomial_nch_ij particles.
//
    **********************************************************************


double NBinomial_nch_ij;
double nhbar_ij;

//P_NBD(NBinomial_n,NBinomial_nbar,NBinomial_k)
//NOTE: n is equivalent to x in
//the distribution used in Negative_Binomial_pdf() definitions

//Random multiplicity obtained from each nucleon-nucleon interaction
//Using Reference Z.Phys.C 43,357(1989)
NBinomial_nch_ij= Negative_Binomial_pdf(NBinomial_nbar,NBinomial_k);
//Average hard multilpicity obtained from NBinomial_nch_ij
//Reference: Lanny Ray, xrXiv:1406.2736v1

nhbar_ij= Alpha_for_nsh_i*NBinomial_nch_ij*NBinomial_nch_ij/(
    Epsilon_for_nsh_i*DeltaEta
      + Alpha_for_nsh_i*NBinomial_nch_ij);
```

```
//Hard multilpicity obtained from average nhbar_ij
//Reference: Lanny Ray, xrXiv:1406.2736v1
nh_i_From_nch_ij= random.Poisson(nhbar_ij);

//Hard multipilicity is integer. So convert to integer
nh_i_From_nch_ij= TMath::Nint(nh_i_From_nch_ij);// Hard Multiplicity

//Get total hard multiplicity by adding multiplicity in each
//nucleuon nucleon interactions
Tot_nh_i_From_nch_ij+=nh_i_From_nch_ij; //Total Hard Multipliciyy


//
    **************************************************************************

//Negative Binomial Distribution, each nucleon collision obtain
//NBinomial_nch_i particles. Soft particles are assumed to be
//produced produced only from only one of binary collisiof for each
    projectile nucleon
//
    **************************************************************************

Random_BinaryColliding_nucleon= NumPart_inA_perPart_inB;
//Temporary. Actually I want it to be random.
// Number shold be at least one out of possible partipicipants in
// Target nucleus for each nucleon in projectile nucleon.

if(NumPart_inA_perPart_inB==1){
//I want it to be from the A nucleon which is closest to b while b
    approaches
//Random multiplicity obtained from each nucleon-nucleon interaction
//Using Reference Z.Phys.C 43,357(1989)
NBinomial_nch_i= Negative_Binomial_pdf(NBinomial_nbar,NBinomial_k);
nch_i= TMath::Nint(NBinomial_nch_i);

//Soft multilpicity obtained from NBinomial_nch_ij
//Reference: Lanny Ray, xrXiv:1406.2736v1
```

```
    ns_i_From_nch_i= nch_i/(1.0+ Alpha_for_nsh_i*nch_i/(Epsilon_for_nsh_i*
        DeltaEta));
    ns_i_From_nch_i= TMath::Nint(ns_i_From_nch_i);
    //It gives nearest integer from the doulble value.
    //ns_i_From_nch_i is not much different from NBinomial_nch_i

    //Use NBinomial_nch_i and ns_i_From_nch_i and Get v2 . Use a
    //function.

    if(nch_i && ns_i_From_nch_i){ //If both are no zero
      Tot_soft_ns_i +=nch_i; //Adding to get total soft multiplicity
      SoftPlusHardTrkID+=nch_i; //Adding to get total soft and hard
          multiplicity

      //Here I want to find out Quadrupole in each nucleon-nucleon
      // collision .
      //Reference: Lanny Ray, xrXiv:1406.2736v1

      double AQ_using_nsi_nchi= Get_AQ_using_nsi_nchi(nch_i,
          ns_i_From_nch_i, DeltaEta);
      double V2= Get_V2(nch_i, DeltaEta, AQ_using_nsi_nchi);
      Get_phi_ij_from_dN_over_dPhi_Dist(outfile,event,nch_i,V2,
                  NucleonPair_Psi[NucB][NucA]);
      //NucleonPair_Psi[NucB][NucA]= ij angle.

      PairWisePsihist->Fill(NucleonPair_Psi[NucB][NucA]);
    }
  }
  }
    }
    if(Tot_nh_i_From_nch_ij){
      Tot_hard_ns_i+=Tot_nh_i_From_nch_ij;
      SoftPlusHardTrkID+=Tot_nh_i_From_nch_ij;
      Get_Random_phi_ij(outfile,event, Tot_nh_i_From_nch_ij);
    }

    delete [] Max_z_cor_A;
```

```
}

if(SoftPlusHardTrkID)
{
  HardperNucleus_ns_i->Fill(Tot_hard_ns_i);
  SoftperNucleus_ns_i->Fill(Tot_soft_ns_i);
  SoftPlusHardperNucleus_ns_ij->Fill(SoftPlusHardTrkID);
  SoftPlusHard14->Fill(pow(SoftPlusHardTrkID,0.25));
}

//
    ************************************************************************

// Compute Npartevent
//
    ************************************************************************

//Strategy: Project  NpartCounter in X and Y axis respectively:
//If the projection bin has at least an entry, one of the parton in
//respected axis nucleus is
//participated in collison otherwise not.

TH1D * projectx= NpartCounter->ProjectionX();
TH1D * projecty= NpartCounter->ProjectionY();
for(int i=1;i<=NoofNucleons_A;i++){
  if (projectx->GetBinContent(i)>=1){
    Npartevent+=1;
  }
  else{
    Nspecevent+=1;
  }
}
for(int i=1;i<=NoofNucleons_B;i++){
  if (projecty->GetBinContent(i)>=1){
    Npartevent+=1;
  }
  else{
```

```
    Nspecevent+=1;
  }
}
delete NpartCounter;
//
    ************************************************************************


 if(Npartevent>0){
  //Means: if there is collision between at least two nucleons.

 //
     ************************************************************************

 // Calculate Nchbar:
 //
     ************************************************************************

 Nchbar= NbarChofb(npp,Kharjeev_x, Npartevent,Nbinevent);
 // Nchbar= TMath::Nint(Nchbar);//Nchbar is an integer ?. Average
     should
 // not necessarily be an integer.
 //
 // This number should be statistically equal to SoftPlusHardTrkID
 // calculated above. In above method it is calculated using slightly
 // different approach. It is to note that the vaplue of Kharjeev_x
     makes
 // difference in the value we get in Nchbar value.
 //
     ************************************************************************


 //
     ************************************************************************

 //Sample Nch, It is a Poisson Random Number with mean Nchbar
 Nch= random.Poisson(Nchbar);
```

```
//
    ************************************************************************


if(Nch==0){
  event+=0;
};
//else{
if(Nch!=0){
event+=1;
Nch14= pow(Nch,1.0/4.0);
Nbin16 = pow(Nbinevent,1.0/6.0);
Npart14 = pow(Npartevent,1.0/4.0);
//
    ************************************************************************


//Fill  Nch and Nch14
Nchhist->Fill(Nch);
Nch14hist->Fill(Nch14);


//Fill  Nbinevent
Nbinhist->Fill(Nbinevent);
Nbin16hist->Fill(Nbin16);
//Fill  Npartevent
Nparthist->Fill(Npartevent);
Npart14hist->Fill(Npart14);

//Fill  b and psi  histos
bhist->Fill(b);
psihist ->Fill(psi);

bVersusNpart->Fill(b,Npartevent);
bVersusNBin->Fill(b,Nbinevent);
}
}
```

```
if(event%100==0)cout<<"Completed "<<event<< " events. "<<endl;

} while (event <=nevents); //Event do loop ends


//Comment if no Histogram is to be plotted

can->cd(1);
Nparthist->Draw();

can->cd(2);
Npart14hist->Draw();

can->cd(3);
//bVersusNpart->Draw();
SoftPlusHardperNucleus_ns_ij->Draw();

can->cd(4);
//bVersusNBin->Draw();
SoftPlusHard14->Draw();

can->cd(5);
bhist->Draw();

can->cd(6);
Nbinhist->Draw();

can->cd(7);
Nbin16hist->Draw();

can->cd(8);
Nchhist->Draw();

can->cd(9);
Nch14hist->Draw();
can->cd(10);
//   NegBinomial_nch_i->Draw();
```

```
can->cd(11);
SoftperNucleus_ns_i->Draw();
can->cd(12);
HardperNucleus_ns_i->Draw();
/*
    can->cd(1);
// bhist->Draw();
gStyle->SetOptStat(0);
PosProfile_B->Draw();
PosProfile_B->SetTitle("");
PosProfile_B->GetXaxis()->SetTitle("r (fm)");
// PosProfile_B->SetLineWidth(1.2);
PosProfile_B->SetTitleSize(0.06,"xy");
PosProfile_B->SetLabelSize(0.05,"xy");
PosProfile_B->SetTitleOffset(0.76,"xy");
PosProfile_B->GetXaxis()->CenterTitle();
PosProfile_B->GetYaxis()->CenterTitle();
PosProfile_B->GetYaxis()->SetTitle("dN/dr");
PosProfile_B->Scale(1.0/nevents);
can->cd(2);
ThetaProfile_B->SetTitle("");
ThetaProfile_B->GetXaxis()->SetTitle("#theta (Rad.)");
ThetaProfile_B->GetYaxis()->SetTitle("dN/d#theta");
ThetaProfile_B->SetTitleSize(0.06,"xy");
ThetaProfile_B->SetLabelSize(0.05,"xy");
ThetaProfile_B->SetTitleOffset(0.76,"xy");
ThetaProfile_B->GetXaxis()->CenterTitle();
ThetaProfile_B->GetYaxis()->CenterTitle();
// ThetaProfile_B->SetLineWidth(1.2);
ThetaProfile_B->Scale(1.0/nevents);
ThetaProfile_B->Draw();
// can->cd(6);
// Nparthist->Draw();
//   can->cd(7);
can->cd(3);
PhiProfile_B->SetTitle("");
PhiProfile_B->GetXaxis()->SetTitle("#phi (Rad.)");
```

```
PhiProfile_B->GetYaxis()->SetTitle("dN/d#phi");
PhiProfile_B->Scale(1.0/nevents);
PhiProfile_B->SetTitleSize(0.06,"xy");
PhiProfile_B->SetLabelSize(0.05,"xy");
PhiProfile_B->SetTitleOffset(0.76,"xy");
PhiProfile_B->GetXaxis()->CenterTitle();
PhiProfile_B->GetYaxis()->CenterTitle();
// PhiProfile_B->SetLineWidth(1.2);
PhiProfile_B->Draw();

// Npart14hist->Draw();
//PhiProfile_B->Draw();
 */ /* can->cd(8);
        Nbinhist->Draw();
        can->cd(9);
        Nbin16hist->Draw();
        can->cd(10);
//DistanceAtoB->Draw();
Nchhist->Draw();
can->cd(11);
Nch14hist->Draw();

can->cd(12);
DistanceAtoB->Draw();


//can->SaveAs("QAhistos.gif");


can->cd(3);

PairWisePsihist->Draw();

can->cd(4);
SoftPlusHardperNucleus_ns_ij->Draw();
//NegBinomial_nch_i->Draw();
//SoftperNucleus_ns_i->Draw();
```

```
can->cd(5);
//z_cor_A_Hist->Draw("text");
//PartID_inA_perPart_inB->Draw("text45");
can->Update();
SoftPlusHard14->Draw();

can->cd(2);
TH2D *projxy_B= (TH2D*)NucleusB->Project3D("xy");
projxy_B->Draw();
projxy_B->SetTitle("");
projxy_B->SetMarkerColor(2);
projxy_B->SetMarkerSize(2.0);
projxy_B->SetMarkerStyle(8);
projxy_B->GetXaxis()->SetTitle("X (fm)");
projxy_B->GetXaxis()->SetTitleOffset(0.75);
projxy_B->GetXaxis()->SetTitleSize(0.06);
projxy_B->GetXaxis()->SetTitleColor(1);

projxy_B->GetYaxis()->SetTitle("Y (fm)");
projxy_B->GetYaxis()->SetTitleOffset(0.75);
projxy_B->GetYaxis()->SetTitleSize(0.06);
projxy_B->GetYaxis()->SetTitleColor(1);

TH2D *projxy_A= (TH2D*)NucleusA->Project3D("xy");

projxy_A->Draw("same");
projxy_A->SetMarkerColor(4);
projxy_A->SetMarkerSize(2.0);
projxy_A->SetMarkerStyle(4);

gStyle->SetOptStat(0);
double degpsi = psi*180/pi;
TString str1;
str1.Form("#Psi = %1.2f^{#circ} \n",degpsi);
TString str2;
str2.Form("b = %1.2f fm\n",b);
TString str3;
```

```
str3.Form("N_{binary} = %d\n", Nbinevent);
TString str4;
TString str5;
str5.Form("");

str4.Form("N_{participant} = %d",Npartevent);
TLegend *leg = new TLegend(0.1,0.62,0.28,0.899);
leg->AddEntry(projxy_B,"Projectile N","p");
leg->AddEntry(projxy_A,"Target N","p");
leg->AddEntry((TObject*)0, str1,"");
leg->AddEntry((TObject*)0, str2,"");
leg->AddEntry((TObject*)0, str3,"");
leg->AddEntry((TObject*)0, str4,"");
leg->AddEntry((TObject*)0, str5,"");
leg->SetFillColor(0);
leg->Draw();

NucleusB->Draw();
NucleusB->SetMarkerColor(2);
NucleusB->SetMarkerSize(0.5);
NucleusB->SetMarkerStyle(8);
NucleusB->GetXaxis()->SetTitle("X");
NucleusB->GetXaxis()->SetTitleSize(0.06);
NucleusB->GetXaxis()->SetTitleColor(4);

NucleusB->GetYaxis()->SetTitle("Y");
NucleusB->GetYaxis()->SetTitleSize(0.06);
NucleusB->GetYaxis()->SetTitleColor(4);

NucleusB->GetZaxis()->SetTitle("Z");
NucleusB->SetTitle("Azimuthal Projection of Nuclear Collision");

gPad->SetTheta(90);
gPad->SetPhi(10);
NucleusA->Draw("same");
NucleusA->SetMarkerColor(4);
NucleusA->SetMarkerSize(0.5);
```

```
NucleusA->SetMarkerStyle(8);

TPolyLine3D *pl3d2 = new TPolyLine3D(150);
pl3d2->SetPoint(30, 40, -15, 8);
// pl3d2->SetPoint(10, 10, -15, 8);
pl3d2->SetLineWidth(5);
pl3d2->SetLineColor(7);
pl3d2->Draw("same");

gStyle->SetOptStat(0);

TString str1;
str1.Form("#Psi = %1.2f\n",psi);
TString str2;
str2.Form("b = %1.2f\n",b);
TString str3;
str3.Form("N_{binary} = %d\n", Nbinevent);
TString str4;
TString str5;
str5.Form("");

str4.Form("N_{participant} = %d",Npartevent);
TLegend *leg = new TLegend(0.75,0.62,0.92,0.899);
leg->AddEntry(NucleusB,"Projectile N","p");
leg->AddEntry(NucleusA,"Target N","p");
leg->AddEntry((TObject*)0, str1,"");
leg->AddEntry((TObject*)0, str2,"");
leg->AddEntry((TObject*)0, str3,"");
leg->AddEntry((TObject*)0, str4,"");
leg->AddEntry((TObject*)0, str5,"");
leg->SetFillColor(0);
leg->Draw();
*/

can->SaveAs("mcgGlauberEventTracksdist.png");
can->SaveAs("mcgGlauberEventTracksdist.eps");
can->SaveAs("mcgGlauberEventTracksdist.pdf");
```

```
can->SaveAs("mcgGlauberEventTracksdist.root");

TTimeStamp TotSecnd;
int FinalTime= TotSecnd.GetSec();
cout<<"Total Time to run "<<FinalTime-startTime <<" seconds for "
    <<nevents
        <<" events @ the rate of "
        <<(float )nevents/(FinalTime-startTime)
        <<" events per second."<<endl;
cout<<"At this rate 1 Million events would take "
        << (float)(FinalTime-startTime)/nevents *1000000<<" seconds
            or "
        << (float)(FinalTime-startTime)/nevents *1000000/60 <<"
            minutes or " <<
        (float)(FinalTime-startTime)/nevents *1000000/(60*60)<<"
            hours. "<< endl;

//Free memory  .............

delete  NucleusA;
delete  NucleusB;
delete  PosProfile_B;
delete  ThetaProfile_B;
delete  PhiProfile_B;
delete  PairWisePsihist;

delete  bhist;
delete  psihist;
delete  Nbinhist;
delete  Nparthist;
delete  Nchhist;
delete  Nch14hist;
delete  Npart14hist;
delete  Nbin16hist;

delete  DistanceAtoB;
delete  L_DistanceAtoB;
```

```
//delete NegBinomial_nch_i;
delete HardperNucleus_ns_i;
delete SoftperNucleus_ns_i;
delete SoftPlusHardperNucleus_ns_ij;
delete SoftPlusHard14;

delete RandomPhihist;

return 0;
}


double NchtoCentrality(int Nch){
  // For Run4 centralities are give on the basis of multiplicity , Now I
      am
  // taking multiplicity as Nch (Actually Nch>=2 is Nmult);
  //For Run 4
  int Centrality;
  if (Nch <=15)Centrality = 0;
  else if (Nch >15 && Nch<=35)Centrality = 1;
  else if (Nch >35 && Nch<=68)Centrality = 2;
  else if (Nch >68 && Nch<=117)Centrality =3;
  else if (Nch >117 && Nch<= 187)Centrality = 4;
  else if (Nch >187 && Nch<= 281)Centrality = 5;
  else if (Nch >281 && Nch<= 401)Centrality = 6;
  else if (Nch >401 && Nch<= 551)Centrality = 7;
  else if (Nch >551 && Nch<= 739)Centrality = 8;
  else if (Nch >739 && Nch<= 852)Centrality = 9;
  else  Centrality = 10;

  return Centrality;

}

double Get_AQ(int Centrality){
  //AQ is Quadrupole amplitude
  //For Run 4 200 GeV Au+Au Collisions
```

```
  double AQ
      []={0.002,0.011,0.028,0.070,0.136,0.201,0.270,0.268,0.179,0.063,0.001};

  return AQ[Centrality];

}


double Get_AQ_using_nsi_nchi(int nch_i, int ns_i, double DeltaEta){
  //AQ is Quadrupole amplitude
  //For 200 GeV p+p Collisions
  //Using Reference L Ray arXiv:1406.2736v1
  double a_0 = -0.000267;
  double a_1 = 0.00048;
  double a_2 = 0.0000243;

  return ns_i/nch_i*(a_0 + a_1*(ns_i/DeltaEta) + a_2*(ns_i/DeltaEta)*(ns_i
      /DeltaEta));

}


double Get_V2(int Nch, double DeltaEta, double AQ){
  const double pi = TMath::Pi();
  double V2 = sqrt(AQ*2*pi*DeltaEta/Nch);
  return V2;

}


//dN_Over_dPhi = Nch/(2pi)(1+2*V2*cos2(phi-psi)
TF1 *f2 = new TF1("f2","(1.0+2.0*[0]*cos(2.0*(x-[1])))",0.0,twopi);
double Get_phi_ij_from_dN_over_dPhi_Dist(ofstream &outfile, int event,int
    Nch,
            double V2, double psi){
  const double twopi = 2*TMath::Pi();
```

```
  //What happens when V2 is multiple of the regular V2 value? It other
      part
  //of code is good, there should be higher correlation of delta Phi.
  V2*=1.0;

  //  TF1 *f2 = new TF1("f2","(1.0+2.0*[0]*cos(2.0*(x-[1])))",0.0,twopi);
  f2->SetParameters(V2,psi);// [0]= V2, [1] = [psi], x= phi
  for (int i=1;i<=Nch;i++){
    double Sample_Phi = f2->GetRandom();
    //    cout<<"EventID\t"<<event <<"\tNch\t"<<Nch<<"\tsoft\t
        "<<i<<"\t_ij\t"<<Sample_Phi<<endl;
     outfile <<event <<"\t"<<Sample_Phi<<endl;

  }
  //  delete [] f2;
  return 0;
}
//I believe this random1 is called only once in the code. If it is called
//repeatedly, the number generated might not be random as expected.
TRandom random1(0);
double Get_Random_phi_ij(ofstream &outfile, int event, int nh_ij){
  //Here it is assumed that the hard particle are scattered randomly in
      twopi
  const double twopi = 2*TMath::Pi();
  for (int i=1;i<=nh_ij;i++){
    double Sample_Phi = twopi*random1.Rndm();
     outfile <<event <<"\t"<<Sample_Phi<<endl;
  }
  return 0;
}
//Distribution of N- Nucleons of "Type A" Nucleus

//The distribution of the centers of nucleons in the nuclear
//ground state , \rho_{pt,m}(r) are given by Woods-Saxon distributions
//  (r)= 0 /[1+exp(r-c)/z)] //For 1-D case
//  (r)~ x [1+exp(r-c)/z)] //For 1-3D case if collision happens along Z
    dir.
```

```
//  (r) = Nuclear density at density r
// 0 = Nuclear density at center of Nucleus
//c = Nuclear radius
//z= skin depth
//This model assumes that the Nucleus is spherical
//So the w parameter of Woods-Saxon distribution is assumed
//to be 0.

//Here NoofNucleons_A is 197 for gold
//cNucleon is c_A  Woods-Saxon parameter of A Nucleon
//zNucleon is z_A  Woods-Saxon parameter of A Nucleon

TF1 *f1 = new TF1("f1","x*x*1.0/(1.0+exp((x-6.43)/0.568))",0.0,10.0);
//Defined on the top
double RadialPos_of_Nucleons(){
  /*    e1 = exp((r-cNucleon)/zNuceon);
   *   rhor ~  x /(1.0+e1)
   *   x = radius (r)
   *   TF1 *f1 = new TF1("f1","x*x*1.0/(1.0+exp((x-6.43)/0.568))",0,10)
       ;
   *   It it better to keep it out side the function to save memory and
   *   computation time. If we keep it inside this function the same
       function
   *  is to be traw each time in the loop.
   */

  return f1->GetRandom();
}
double NbarChofb(double npp, double Kharjeev_x, double Npart,
    double Nbin){
  //   Compute the average multiplicity at impact parameter b using
  //   Kharzeev and Nardi, Phys. Lett. B 507, 121 (2001) model

  return (1.0 - Kharjeev_x)*npp*Npart/2.0 + Kharjeev_x*npp*Nbin;
}
double GetsigmaNN(double b,double sigma0,double beta,double b0,
    double zb){
```

```
  //   Compute impact parameter dependent NN inelastic total cross
       section:
  //   Variable Type Declarations:

  double e1 = exp((b - b0)/zb);
  return  sigma0 + beta/(1.0 + e1);
  //This is return value for sigmaNN. For Now beta =0. beta is for
       systematic check.
}

double GetDistAtoB(double r1,double theta1,double phi1,double r2,
       double theta2,double phi2,double b,double psi){
  //           Compute impact parameter magnitude for local nuclear
       coordinates
  //           1,2 for nuclei offset along the x-axis by impact parameter
       b.
  //           Angles are in radians.
  //           Variable Type Declarations:
  double x1,y1,x2,y2;
  x1 = r1*sin(theta1)*cos(phi1);
  y1 = r1*sin(theta1)*sin(phi1);

  x2= r2*sin(theta2)*cos(phi2)+b*cos(psi); //Offset is along any direction
       in XY plane
  y2= r2*sin(theta2)*sin(phi2)+b*sin(psi); //Offset is along any direction
       in XY plane

  return (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2);
  //  return pow((x1-x2),2) + pow((y1-y2),2);
  // return sqrt(pow((x1-x2),2) + pow((y1-y2),2));
}

double LannyDist(double r1,double theta1,double phi1,double r2,
       double theta2,double phi2,double b){
  double x1,y1,x2,y2;
  double SDiff;
```

```
  x1 = r1*sin(theta1);
  y1 = x1*sin(phi1);
  x1 = x1*cos(phi1);
  x2 = r2*sin(theta2);
  y2 = x2*sin(phi2);
  x2 = x2*cos(phi2) + b;
  SDiff = sqrt(pow((x1-x2),2) + pow((y1-y2),2));
  return SDiff;
}

//Copied from ROOT : PdfFuncMathCore.cxx


// xn  in Note, nbar is  first  parameter, k is  second parameter, I assume
    n
//could go upto 20.
TF1 *NBinfit1= new TF1("NBinfit1","(x+1)*(TMath::Power
    (([1]/[2]/(1+[1]/[2])),x))
          *(TMath::Power((1+([1]/[2])),-[2]))",0,20);
//When I use factorial,  root  forces  the argement to be integer.  That
    creates
//some artifacts.  So I  have  simplified  the  relation  using [2]=2 and some
    basic
//algebra.
double Negative_Binomial_pdf(double nbar, unsigned int k) {
  //NBinfit1->SetParameter(0,1.);// normalization constant
  NBinfit1->SetParameter(1,nbar); // mean multiplicity
  NBinfit1->SetParameter(2,k); // k parameter
  return NBinfit1->GetRandom();

}
```

```
#ifndef MCGSimulation_H
#define MCGSimulation_H
#include<fstream>
//Woods-Saxon Parameters:
//Note: parametesr are different  for  diffent  Nucleus
```

```
// For Au−Au:
const int NoofNucleons_A =197; //Number of Nucleons of Type A Nucleus
const int NoofNucleons_B =197; //Number of Nucleons of Type A Nucleus
Double_t c_A= 6.43; //For Type A Nucleus
Double_t c_B= 6.43; // For Type B Nucleus
Double_t z_A= 0.568; //For Tyep A Nucleus
Double_t z_B= 0.568; //For Type B Nucleus
Double_t rho0_A = 1.0; //This is normalized for now.
Double_t rho0_B = 1.0; //This is normalized for now.
Double_t rho0 =1.0; //This is normalized common factor
Double_t r; //Position of Nucleon from Center.
Double_t e1; // An algebraic factor
Double_t delta_r = 0.1; // This is delta r
Double_t rmax= 10;// Maximum value of r
Double_t nrpts = rmax/delta_r +1.0000; //Number of delta_r for integration
Double_t rhor; //Nuclear Density at distance r

Double_t b; //Impact parameter
Double_t del_b=1.0;
Double_t bmax = 20;

Double_t psi;//This is random collision angle along phi direction

Int_t Npartevent; //Npart
Int_t Nbinevent; //Nbinary
Int_t Nspecevent;//Nspectators
Int_t Awounded[NoofNucleons_A*NoofNucleons_B];
Int_t Bwounded[NoofNucleons_A*NoofNucleons_B];



Double_t sigmaNNofb; //Cross section for NN collision
Double_t sigma_inel_0 = 4.17; //For inelastic cross section
Double_t beta=0;
Double_t b0=6.00;
```

```
Double_t zbthick= 0.50;
Double_t sigmabCut;

Double_t DistAtoB;


Double_t Nchbar;//Average Charged Multiplicity
Int_t  Nch;
Double_t Nch14;
Double_t Npart14;
Double_t Nbin16;
Double_t Kharjeev_x=0.13; //Kharjeev and Nardis x parameter, Ref. J.Phys.
     G:Nucl. Part. Phys. 35(2008) L Ray Daugherity
Double_t Nchbar_pp= 2.43; //Average multiplicity in pp collisions
Double_t DeltaEta =1; //
Double_t npp = Nchbar_pp/DeltaEta;
Int_t  event;
Int_t  nevents= 40000; //Total number of collision events  .In stampede
    10000 events takes about 1 hour.



#endif
```

Bibliography

[1] D.J. Prindle and T. A. Trainor. The equivalence of fluctuation scale dependence and autocorrelations. *Journal of Physics: Conference Series*, 27(118):10, 2005.

[2] L. Ray and M. Daugherity. Applicability of Monte Carlo Glauber models to relativistic heavy ion collision data. *J. Phys. G: Nucl. Part. Phys.*, 35, 2008.

[3] Michael L. et. al. Glauber Modeling in High-Energy Nuclear Collisions. *Annu. Rev. Nucl. Part. Sci.*, 57(205):43, 2007.

[4] Kharzeev D. and Nardi M. Hadron production in nuclear collisions at RHIC and high-density QCD. *Phys. Lett. B*, 507(121):8, 2001.

[5] R.E. Ansorge et.al. Charged particle multiplicity distributions at 200 and 900 GeV c.m. energy. *Z. Phys. C*, 43(357):18, 1989.

[6] R. L. Ray. Azimuthal quadrupole correlation from gluon interference in 200 gev and 7 tev p p collisions. *PHYSICAL REVIEW D*, 90(054013):10, 2014.

[7] T. A. TRAINOR. The rhic azimuth quadrupole: perfect liquid or gluonic radiation? *Modern Physics Letters A*, 23(8):20, 2008.

[8] A. M. Poskanzer and S. A. Voloshin. Methods for analyzing anisotropic flow in relativistic nuclear collisions. *PHYSICAL REVIEW C*, 58(3):8, 1998.

Vita

Prabhat Bhattarai, the son of Dharmananda Bhattarai and Sita Devi Bhattarai, was born in Nepal. He graduated from University of Minnesota Duluth with M.S. in Physics in 2010. He entered the University of Texas at Austin in fall 2010 and received doctoral candidacy in Physics in 2013.

Permanent address: Department of Statistics and Data Sciences, The University of Texas at Austin, Austin, TX 78712, USA

This report was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.